COMPUTER SCIENCE

HIGHER SECONDARY FIRST YEAR

VOLUME II - CHAPTER 10

PROBLEM SOLVING TECHNIQUES AND C PROGRAMMING

1,2,3 & 5 MARKS

S.LAWRENCE CHRISTOPHER, M.C.A., B.Ed.,

LECTURER IN COMPUTER SCIENCE

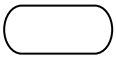
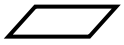
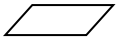
PONDICHERRY

CELL NO: 9486385585

CHAPTER 10

PROBLEM SOLVING TECHNIQUES AND C PROGRAMMING

Multiple Choice Questions And Answers

1. The_____ we do is independent of the computer language we use.
   a) developing b) executing **c) computing** d) programming
2. In the computer languages every statement must be written _____
   a) precisely b) randomly c) manually d) approximately
3. _____languages are in between the natural languages and the computer languages.
   a) system b) translator **c) intermediate** d) machine
4. Which of the following is intermediate language?
   a) flowchart b) pseudocode c) english **d) both a and b**
5. The flows of computational paths are depicted as a picture, it is called as _____
   a) flowchart b) pseudocode c) coding d) algorithm
6. Which of the symbol is used for input/output in flowcharts?
   a)  b)  **c)**  d) 
7. Which of the symbol is used for start/end in flowcharts?
   a)  **b)**  **c)**  d) 
8. Which of the symbol is used as connector in flowcharts?
   a)  b)  c)  **d)** 
9. In flowchart  symbol depicts _____
   a) input **b) decision** c) output d) connector
10. How many fundamental control structures are there?
    a) two b) four **c) three** d) many
11. The fundamental control structure usually the calculations are done one after another is called__
    a) order b) structure c) program **d) sequence**
12. In _____ branching branches to one of the two available paths depending on the answer.
    a) two-way b) three-way c) multi-way d) sequence
13. ____ is depicted by a diagonal shaped box.
    a) process b) output c) input **d) two-way branching**
14. Depending on the answer, we may have to make different set of computations, by going through different paths is called _____ branching
    a) two-way b) three-way **c) multi-way** d) sequence
15. Repeating a set of actions again and again is called as _____
    a) structure b) sequence **c) iteration** d) two-way branching
16. _____ variable is used to keep track of the count of the number of times the actions are performed.
    a) integer b) control c) index **d) either b or c**
17. Control variable is also called as _____ variable.
    a) process b)condition c) input **d) index**

18. The index variable should be given as _____
a) character **b) integer** c) sign d) symbol
19. The current value in the index variable should be compared with the _____ to decide whether more iteration is required.
a) computation b) output **c) final value** d) control value
20. Instead of using flow chart, _____can be used to represent a procedure for doing something.
a) coding chart **b) pseudocode** c) program d) algorithm
21. _____ is in-between English and the high-level computer languages.
a) flowchart **b) pseudocode** c) program d) algorithm
22. A method of checking flowchart or pseudo code is called as _____
a) verification b) compilation c) algorithm **d) walkthrough**
23. _____ written with the specific syntax rules of a particular language.
a) flowchart b) pseudocode **c) program** d) algorithm
24. A flow chart is drawn _____ writing a program.
a) while **b)before** c) after d) either b or c
25. An _____is a procedure a finite number of steps.
a) flowchart b) pseudocode c) program **d) algorithm**
26. Examples of object oriented approach are _____
a) C, Cobol b) Pascal, C c) Fortran, Pascal **d) C++, Java**
27. C programming language was developed by _____
a) Ken Thompson b) Tannenbaum **c) Dennis Ritchie** d) Linus Torvalds
28. C programming language was developed at _____ laboratory
a) AT & T Bell b) IBM c) CERN d) Micro system
29. C language was designed originally as a language to be used with ____ operating system
a) Linux **b) Unix** c) Mac d) Windows
30. _____ language is a general-purpose language.
a) Linux **b) C** c) C++ d) BASIC
31. In C language, the basic types of elements are collectively known as _____
a) keywords b) variables **c) tokens** d) data types
32. ____is a source program text that the compiler does not break down into component elements.
a) keywords b) algorithm **c) tokens** d) pseudo code
33. Which of the following is a token?
a) constants b) identifiers c) operators **d) all the above**
34. The value of a _____ cannot be modified.
a) data b) variable **c) constant** d) string
35. A non-numeric data can be called as a_____.
a) character **b) literal** c) string d) floating-point
36. Numeric constants are of _____ types.
a) two **b) three** c) four d) five
37. _____ constant comprises of the digits 0 to 9.
a) decimal b) floating-point c) character **d) integer**
38. The hexadecimal integer constant begins with the letters _____
a) 0x b) 0X c) X0 **d) either a or b**
39. _____are not allowed in an integer constant.
a) hexadecimals b) octal numbers c) negative numbers **d) Special characters**
40. _____ constant is a signed real number.
a) decimal b) negative c) character **d) floating point**
41. In floating-point constant, _____ is represented in powers of 10 in decimal system
a) integer **b) exponent** c) decimal d) fraction

42. Which of the following is of numeric or non-numeric type?
a) Constant b) Variable c) Keyword d) Identifier
43. A non-numeric data can be called a _____
a) Constant b) Variable **c) Literal** d) Integer
44. An integer constant has a base _____
a) 16 **b) 10** c) 8 d) 2
45. **58.64** is represented in exponent form as _____
a) 5864 X 10^1 b) 5.864 X 10^{-1} **c) 5.864 X 10^1** d) 5.864 X 10^2
46. The letter ____ is used to represent the floating-point constant in exponent form.
a) e b) E c) F **d) either a or b**
47. **58.64** is represented in exponent form as ____
a) 5864 X 10^{-2}b) 0.5864 X 10^2 c) 5.864 X 10^1 **d) all of these**
48. **58.64** is represented in exponent form as ____
a) 5864E-2 b) 0.5864e2 c) 5.864 E1 **d) all of these**
49. _____is a letter, numeral or special symbol, which can be handled by the computer system.
a)data **b) character** c) input d) token
50. The characters used in C language are grouped into _____ classes.
a) three b) two c) four d) many
51. Which of the following is an invalid constant?
a) '+' **b) a** c) '1' d) 'a'
52. Character combinations consisting of a backslash \ followed by a letter are called _____
a) special symbols b) constants c)strings **d) escape sequences**
53. Which of the following escape character is used for new line?
a) '\a' b) '/a' **c) '\n'** d) '/n'
54. Which of the following escape character is used for null character?
a) '\0' b) '/0' c) '\n' d) '/n'
55. _____is a sequence of characters from the system's character set, enclosed in double quotes.
a) string literal b) string constant **c) either a and b** d) character constant
56. By default, _____is assumed as the last character in a string literal.
a) '\0' b) '/0' c) '\n' d) '/n'
57. _____ are the names that are to be given to the variables and functions.
a) constants b) strings c) data types **d) identifiers**
58. The length of a variable may vary from one character to ____ characters
a) 8 **b) 32** c) 28 d) 30
59. The valid variable name in C program is _____
a) x_value b) a123 c) length **d) all of these**
60. Which of the following is an valid variable name in C program?
a) x_value b) a123 c) length **d) 1abc**
61. Which one of the following cannot be used as an identifier?
a) alphabets b) numbers **c) Keywords** d) underscore
62. Which one of the following has a special meaning in C?
a) Identifiers b) Constants **c) Keywords** d) Punctuators
63. _____can be defined as the raw information input to the computer
a) data b) program c) code d) information
64. There are _____ numeric data types available in C language.
a) three b) two c) four d) five
65. An integer requires _____ of memory to store its value.
a) 3 bytes **b) 2 bytes** c) 4 bytes d)1 byte

66. A float requires ____ bytes of memory to store its value.
a) 3 bytes b) 2 bytes **c) 4 bytes** d)1 byte\

67. A character requires _____of memory to store its value.
a) 3 bytes b) 2 bytes c) 4 bytes **d)1 byte**

68. The data type **double** occupies _____ in the memory
a) 3 bytes b) 2 bytes **c) 8 bytes** d)1 byte

69. To store a **long** integer value, _____ of memory are required.
a) 1 byte b) 2 bytes c) 8 bytes **d)4 bytes**

70. **Unsigned int** occupies _____ as normal integers.
a) 1 byte **b) 2 bytes** c) 8 bytes d)4 bytes

71. Which one of the following is a derived data type in C?
a) float b) char **c) unsigned** d) int

72. Which one of the following is a derived type from the fundamental primitive types?
a) long b) double c) unsigned **d) all of these**

73. A pointer variable is declared as _____
a) int y; b) int y*; **c) int *y;** d) *int y;

74. Which of the following is an address of operator ?
a) && **b) &** c) * d) #

75. Identify the operators, which are associated with pointer.
a) & and * b) & and ! c) * and @ d) * and &&

76. Which of the following is an indirection operator ?
a) @ b) & **c) *** d) #

77. Both **address of** and **indirection** operators are _____ operators.
a) ternary b) logical c) binary **d) unary**

78. To obtain the address of the variable, we have to use the _____ operator.
a) **address of (&)** b) indirection(*) c) and (&&) d) size of

79. To retrieve the value of a variable through the pointer variable we can use the _____operator.
a) address of (&) **b) indirection(*)** c) and (&&) d) size of

Read the following C program statements and Answer Q. No 80 – 82 .

```
int x;
int * y;
x =10;
y=&x;
```

The address of the variable **x** is **948**

80. Value stored in **y** is _____
a) 1 b) 10 **c) 948** d) nothing

81. Value stored in x is _____
a) 1 **b) 10** c) 948 d) nothing

82. Value stored in ***y** is _____
a) 1 **b) 10** c) 948 d) 0

83. _____is defined as a symbol that specifies an operation to be performed.
a) data type b) operand c) variable **d) operator**

84. The order in which operations are performed is called _____
a) expression b) hierarchy c) order of precedence **d) either b or c**

85. The direction in which operations are carried out is called _____.
a) associativity b) hierarchy c) sequence d) expression

86. There are _____types of operators in C.
a) five b) two **c) three** d) four

87. Order of precedence is high for the ____ operators.
a) unary b) binary c) assignment d) logical

88. _____ operators have only one operand.
a) unary b) binary c) ternary d) logical

89. The order of evaluation (associativity) is from _____
a) top to bottom b) left to right **c) right to left** d) any order

90. Which of the following is not a unary operator?
a) & b) ++ c) -- **d) &&**

91. Which of the following is not a unary operator?
a) + b) ++ c) -- d) !

92. The increment / decrement operator is used to increase or to decrease the current value of a variable by _____
a) 0 **b) 1** c) 2 d) 10

93. _____ increment or decrement operators appear before the operand.
a) positive b) suffix **c) prefix** d) postfix

94. _____ increment or decrement operators appear after the operand.
a) positive b) suffix c) prefix **d) postfix**

95. Binary operators have _____operands.
a) multiple **b) two** c) three d) four

96. All the arithmetic operators observe _____ associativity.
a) top to bottom **b) left to right** c) right to left d) any order

97. Which of the following is an arithmetic operator?
a) + b) * c) % **d) all of these**

98. The statement **5 % 2** gives the result _____
a) 2 **b) 1** c) 5 d) 2.1

99. The relational or Boolean operators are _____ operators
a) unary **b) binary** c) ternary d) assignment

100. Boolean operator is called as a _____ operator
a) unary b) binary c) ternary **d) negation**

101. Which of the following is not a logical operator?
a)! b) || c) && **d) &**

102. _____operator returns TRUE if both of its operands evaluate to TRUE.
a) AND b) OR c) NOT d) IF

103. The expression (10 < 15) && (14 < 23) is always _____
a) >1 b) false **c) true** d) -1

104. Two relational expressions are combined using _____ operator
a) unary b) binary **c) logical** d) arithmetic

105. The symbol represents the logical OR operator is _____
a)! **b) ||** c) && d) &

106. The symbol represents the logical AND operator is _____
a)! b) || **c) &&** d) &

107. _____ operator returns TRUE when one or both of its operands evaluates to TRUE.
a) AND **b) OR** c) NOT d) IF

108. The expression (10 < 15) || (14 > 23) gives _____
a) >1 b) false **c) true** d) -1

109. _____operator assigns the value of the right-hand operand to the left-hand operand
a) assignment b) logical c) relational d) arithmetic
110. _____ is an assignment operator.
a) != b) <> c) == **d) =**
111. What is the result of the expression: **5 * 2 + 8 + (3 – 2) * 5**
a) 250 **b) 23** c) 75 d) 85
112. Ternary operator is also known as _____ operator
a) logical b) boolean **c) conditional** d) unary
113. The symbol used for ternary operator is _____
a) ?: b) :? c) :: d) *
114. _____ symbol is used to represent array index.
a) ( ) b) < > c) { } **d) [ ]**
115. _____ symbol is used to represent a function.
a) ( ) b) < > c) { } d) [ ]
116. _____ symbol is used to cover the body of the function.
a) ( ) b) < > **c) { }** d) [ ]
117. _____ symbol is used to enclose the header file in a preprocessor statement.
a) ( ) **b) < >** c) { } d) [ ]
118. _____ symbol is used as a statement terminator.
a) . b) , **c) ;** d) :
119. _____ is a program used to carry out some small task.
a) function b) coding c) operation d) main
120. When a C program runs, the control is transferred _____ function.
a) printf() **b) main()** c) scanf() d) Main()
121. _____ is called the program's entry point.
a) printf() **b) main()** c) scanf() d) Main()
122. _____ is an example for pre-defined function.
a) printf() b) main() c) scanf() **d) both a and c**
123. _____is a preprocessor directive.
a) #INCLUDE **b) #include** c) stdio.h d) <stdio.h>
124. The pre-defined function **clrscr()** is available in _____
a) stdio.h b) io.h c) coni.h **d) conio.h**
125. Each and every line of a C program can be considered as a
a) coding **b) statement** c) rule d) procedure
126. There are generally _____ types of statements
a) many b) two c) three **d) four**
127. _____statement is used to include the function declaration statements from the specified header files.
a) function header b) declaration **c) preprocessor** d) executable
128. Which of the following is an example for variable declaration statement?
a) a = 10; **b) int a,b,c;** c) main() d) #include <stdio.h>
129. An assignment statement is defined as _____
a) Expression=variable b) Expression==variable
c) Variable==Expression **d) Variable = Expression;**
130. Which of the following is an example for Postfix increment?
a)i++ b) i+ c) ++i d) +i
131. The statement c=a+b is an example of _____ expression.
a) arithmetic b) assignment c) relational d) logical

132. The statement f=d=e is an example of _____ expression.
a) arithmetic **b) assignment** c) relational d) logical

133. What is the output of the following program segment?

```
int x, i;
i = 10;
x = i++;
printf("%d %d\n", x, i);
```

a)10 10 b) 11 10 **c) 10 11** d) 11 11

134. The statement i=i+1 can be written as _____
a)i++ b) ++i c) i+=1 **d) i++, ++i or i+=1**

135. The statement a=b>c is an example of _____ expression.
a) arithmetic b) assignment **c) relational** d) logical

136. _____ is used to display the results on the standard output (screen)
a) print() **b) printf()** c) scanf() d) Printf()

137. The first parameter of the printf() function used to control the output is called _____
a) valid string b) output string c) string **d) control string**

138. What is value of **x** and **i** in the following program segment?

```
int x, i;
i = 10;
x = ++i;
printf("%d %d\n", x, i);
```

a) 10 10 b) 11 10 c) 10 11 **d) 11 11**

139. What will be the output?

```
int x, z;
x = 100;
z = (x==x++);
printf("%d %d", z,x);
```

a) 0 101 b) 100 100 c) 100 101 d) 1 101

140. The parameter is used to format the output for display is called as _____
a) formatting **b) formatting string** c) string d) output string

141. _____is used as a formatting character to display the value of an integer
a) %i b) %f **c) %d** d) %ld

142. The formatting character used to display the value of an **float** type variable is _____
a) %i **b) %f** c) %d d) %ld

143. The formatting character used to display the value of an **char** type variable is _____
a) %s b) %f c) %d **d) %c**

144. If y=10.5, the output of **printf("%f",y)** is
a) 10.5 b) 10.50 c) 10.500 **d) 10.500000**

145. By default, the floating-point values are displayed with ____decimal places of accuracy
a) six b) five c) four d) three

146. _____ function is used to read a value from the keyboard.
a) print() b) scan() **c) scanf()** d) read()

147. The function which calls another function is termed as _____function
a) invoked **b) calling** c) user-defined d) called

148. A function declaration may be called as a _____.
a) function model b) function prototype c) function call **d) either a or b**

149. Functions are invoked by a _____
a) function model b) function prototype **c) function call** d) function definition

150. The function prototype has _____ components.
a) six b) five **c) four** d) three

151. The code written within the curly braces is called as _____
a) function body b) function prototype c) function block **d) either a or c**

152. All variables declared in function definitions are _____variables.
a) local b) global c) file d) function

153. A function's _____are also local variables.
a) data types **b) parameters** c) statements d) declarations

154. _____provide the means for communicating information between the calling function and called function.
a) data types **b) parameters** c) statements d) declarations

155. _____parameters are the parameters defined in the calling function.
a) Local b) Global **c) Actual** d) Formal

156. _____parameters are the parameters defined in the called function.
a) Local b) Global c) Actual **d) Formal**

157. _____ is a last-in - first-out (LIFO) structure
a) queue **b) stack** c) tree d) line

158. In a function, parameters are stored onto a _____
a) queue **b) stack** c) tree d) line

159. In a function, parameters are stored onto a stack from _____
a) left to right b) top to bottom **c) right to left** d) bottom to top

160. _____is an attribute that is associated with the variable.
a) function b) value c) data type **d) Storage class**

161. C Language provides _____storage classes.
a) six b) five **c) four** d) three

162. A variable's storage class is used to determine its _____ and _____
a) name, type b) type, life time c) name, scope d) scope, lifetime

163. _____variables are actually local variables
a) local b) global c) actual d) formal

164. We cannot access the values of the _____ variables outside the function
a) local b) global c) extern d) register

165. What is the scope of the variable **i?**

```
add()
{
int i = 0;
i = i + 1;
}
```

a) register b) global c) extern **d) local**

166. _____ variable retained its value even after execution of the function
a) local **b) static** c) extern d) register

167. _____variables are declared before the main() function
a) local **b) global** c) extern d) register

168. _____ variable can be accessed and modified by all the functions in the program.
a) local **b) global** c) extern d) register

169. The life time of the _____ variable ends only when the entire program execution is over
a) local b) global c) static **d) both b and c**

170. _____variables behave like auto variables.
a) static b) global c) extern **d) register**

171. The value of _____variable is placed in one of the computer's high-speed hardware registers.
a) static b) global c) extern **d) register**

172. _____ variables are used to speed up operations by reducing memory access time.
a) static b) global c) extern **d) register**

173. _____variables have global scope and lifetime is throughout the execution of the program.
a) local b) auto **c) extern** d) register

174. _____statement controls conditional branching
a) if b) output c) input d) function

175. _____statement is the modular replacement of the cumbersome nested if-else structure.
a) if b) multiple-if **c) switch-case** d) if-case

176. _____statement transfers the control out of the switch-case body.
a) continue b) exit c) default **d) break**

177. ____statement is executed if no case is equal to the value of switch-case.
a) continue b) exit **c) default** d) break

178. _____ is a part of a program that comes back and repeats itself as many times as necessary.
a) loop b) function c) control d) if-else

179. In the _____ loop the condition is tested at the entry level.
a) do…while b) if…else c) entry **d) while**

180. In a nested while statement, inner while statement executes _____ than the outer while loop
a) slower **b) faster** c) lower d) later

181. _____loop is a definite repetition loop.
a) for b) while c) do while d) either b or c

182. _____ is a pre-defined function used to read a character at a time from the keyboard
a) read() b) printf () **c) getchar()** d) gets()

183. In _____ statement, the body of the loop is executed at least once whether the condition is true or false
a) for **b) do-while** c) if-else d) while

184. In _____ statement, the condition is tested at the exit level.
a) for **b) do-while** c) if-else d) while

185. _____ is a collection of homogeneous elements of similar data type.
a) array b) loop c) variable d) structure

186. An array declaration specifies the _____of an array and the _____of its elements
a) value, name b) type, value c) name, value **d) name, type**

187. The number of elements of an array must be _____
a) <1 **b) > 0** c) >variable d) < 0

188. There are _____ types of array in C.
a) two b) three c) four d) five

189. The square brackets in an array specify the _____ of elements in the array
a) type b) value **c) number** d) structure

190. The elements of an array are stored in _____ memory locations.
a) order **b) contiguous** c) random d) stack

191. The array elements can be accessed using _____
a) types b) values c) variables d) indices

192. An array index starts from _____ to _____
a) 0 to n-1 b) 0 to n c) 1 to n-1 d) 1 to n
193. How many bytes are allocated by the compiler in the main memory for an array **int a[10]; ?**
a) 10 **b) 20** c) 11 d) 21
194. The address of the first element is represented for **int a[3];** is _____
a) &a[1] b) &a[0] c) a **d) both b and c**
195. Which of the following provide the same value in array?
a) *(&a[0]) b) a[0] c) *a **d) all of these**
196. Which symbol represents "all are one and the same"?
a) <=> b) >= c) == d) (==)
197. The array's name always points to the _____address of the array.
a) middle b) any **c) starting** d) last
198. The starting address of an array is also known as _____ which cannot be modified.
a) base name **b) base address** c) base type d) first address
199. The address is stored in the array name it becomes a _____
a) pointer b) base c) value d) structure
200. _____can be defined as a collection of characters.
a) array **b) string** c) address d) structure
201. The data type associated with the string constant is _____
a) char b) int c) int * **d) char ***
202. _____ header file provides declarations of many string handling functions.
a) STRING.H b) char.h **c) string.h** d) string
203. _____ function is used to find the length of the string.
a) stlen() b) strlength () c) length() **d) strlen()**
204. _____array has been considered as an array of arrays in C language.
a) multi b) two c) single **d) multidimensional**
205. The first dimension in a multi-dimensional array represents the number of _____
a) strings **b) rows** c) columns d) values
206. In a multi-dimensional array, the second dimension represents the number of _____
a) strings b) rows **c) columns** d) values
207. We can access the first element in a multi-dimensional array using _____
a) a[1][0] **b) a[0][0]** c) a[1][1] d) a[0][1]
208. _____are derived data types in C language
a) char b) float c) int **d) structure**
209. Which are commonly used to define records to be stored in files in C?
a) arrays b) data types c) fields **d) structures**
210. A _____is a collection of records.
a) array **b) file** c) field d) structure
211. A _____ is a collection of fields of information.
a) array b) file c) field **d) record**
212. _____ is a homogeneous collection of elements.
a) array b) file c) fields d) structure
213. _____ is a heterogeneous collection of elements.
a) array b) file c) record **d) structure**
214. _____is a keyword, which is used to define a structure.
a) struct b) structure c) int d) void
215. Which operator is used to access the members of a structure?
a) * b) & **c) .** d) =

TWO MARKS QUESTIONS AND ANSWERS

1. What is a Flowchart?

The flows of computational paths are depicted as a picture. It is called a **flow chart**.

2. What is intermediary language? Give examples.

Language which is in between the natural languages and the computer languages is called as an intermediary language. **Examples:** Flowchart, Pseudo code

3. Why we write intermediary language?

- To understand the problem clearly without any ambiguity, we write it in an intermediary language.
- This will be easy to write and understand

4. What are the advantages of flowchart?

- They are Precise. They represent our thoughts exactly.
- It is easy to understand small flowcharts.

5. Give two examples where multi-way branching is more natural than two-way branching.

i) What is the age of a student?
ii) Which alphabet is vowel?
iii) What is the grade of the student?
iv) Whether a number is negative, positive or zero?

6. What are the three types of fundamental control structures?

i) Sequencing
ii) Branching
iii) Iteration

7. What is sequencing?

Sequencing is one of the fundamental control structures. Usually calculations are done one after another, in a sequence.

8. What are the two types of branching?

i) Two-way branching
ii) Multi-way branching

9. What is Walkthrough?

A method of checking the way in which a computer will work using flowchart or pseudo code is called a walkthrough.

10. What is Top-down approach?

- To create a program, the problem should be divided into many smaller problems.
- Results of these sub problems are putting together to get the result for the bigger problem.

11. Give some examples for system software developed by C language.

- Operating systems
- Compilers
- Text processors
- Database management systems

12. Compare Structured Programming and Object oriented Approach.

Structured Programming	Object Oriented Approach
Importance is given to the procedures, not for the data	Importance is given to both procedures and data
Examples: C, Pascal	**Examples:** C++, Java

13. What is a constant?

- A constant is of numeric or non-numeric type.
- It can be a number, a character or a character string that can be used as a value in a program.
- The value of a constant cannot be modified

14. What are the three types of numeric constants?

i) integer constant
ii) floating-point constant
iii) character constant

15. What are the different ways to represent a floating-point constant 58.64?

- 5.864E1 => 5.864 X 10^1 => 58.64
- 5864E-2 => 5864 X 10^{-2} => 58.64
- 0.5864e2 => 0.5864 X 10^2 => 58.64

16. What is String Literal?

- A **string literal** or a **string constant** is a sequence of characters
- It is enclosed in double quotes.
- By default, the null character '\0' is assumed as the last character in a string literal.

Example: "hello"

17. What is meant by identifier?

Identifiers are the names that are to be given to the variables, functions, data types and labels in a program.

18. What are keywords in C?

- Keywords have special meaning in C
- They are reserved words by compiler for specific purposes.
- They cannot be used as identifiers.

Examples: auto break switch do if

19. What is Data?

Data can be defined as the raw information input to the computer.

20. List the fundamental data types in C

- int
- float
- char

21. What are the memory requirements to store the fundamental data types? (OR) How many bytes require by the fundamental data types to store their value?

Data type	Bytes
Character	1
Integer	2
Float	4

22. Write short notes on Derived types in C.

The derived types from the fundamental primitive types are:

- long
- double
- unsigned
- arrays
- pointers

23. How many bytes require by the derived types to store their value?

Data type	Bytes
unsigned int	2
long int	4
double	8

24. How are variables classified in C? Differentiate them.

The variables in C are classified into **ordinary variables** and **pointer variables**.

- Ordinary variable - takes values of its associated type
- Pointer variable - assumes only address as its value

25. List the operators associated with pointers.

There are only two operators associated with pointers:

i) address of (&) operator
ii) indirection (*) operator

26. What are operators?

An operator is defined as a symbol that specifies an operation to be performed. Operators inform the computer what tasks it has to perform as well as the order in which to perform them.

27. What are the types of operators in 'C'?

There are three types of operators in C.

1) Unary operators,
2) Binary operators
3) Ternary operator

28. Define hierarchy.

The order in which operations are performed is called the **order of precedence**. It is also called as **hierarchy.**

29. What is associativity?

The direction in which operations are carried out is called associativity.

30. What is the use of assignment operator?

The assignment operator (=) assigns the value of the right-hand operand to the left-hand operand.

Example: a =10;

31. List the unary operators in C.

Symbol	Type of operation	Associativity
++	Increment	Right to Left
- -	Decrement	
*	Indirection	
&	Address of	
!	Negation (logical NOT)	

32. What are the two forms of increment/decrement operator?

i) Postfix increment or decrement - operators when they appear after the operand.
Example: i++ i - -

ii) Prefix increment or decrement - operators when they appear before the operand.
Example: ++i - -i

33. List the arithmetic assignment operators.

Symbol	Example	Meaning
+=	i += 1	i = i + 1
- =	i - = 1	i = i – 1
*=	i *= 1	i = i * 1
/=	i /= 1	i = i / 1
%=	i %= 1	i = i % 1

34. What is a program?

A program is defined as a set of instructions to be executed sequentially to obtain the desired result.

35. What is a function?

A function is a program, which is being used to carry out some small task. A function may be pre-defined or user-defined.

36. Which is program's entry point? (OR) what is the purpose of main() function?

- The main() function is a user-defined one.
- The user has to define the main() function to provide necessary code.
- When a C program runs, the control is transferred to this function.
- This is called the program's entry point

37. What is an expression?

An expression occurs usually on the right-hand side of an assignment statement. It has a value when it is evaluated.

38. What is a parameter?

A parameter is a data or information passed on to the called function. Parameters are given one after another within the brackets

39. What is preprocessor statement?

- The first line in the program is a preprocessor statement.
- **#include** is a preprocessor directive.
- The preprocessor is a software program that will expand the source code while the program is compiled.
- **Example:** #include <stdio.h>

40. List the types of statements.

There are generally four types of statements. They are:

1) Preprocessor statement
2) Function header statement
3) Declaration statement
4) Executable statement

41. Write short notes on assignment statement.

An assignment statement is defined as:

Variable = Expression;

- A semicolon terminates the assignment statement.
- The value of the expression is assigned to the left hand side variable.
- The '=' sign is the assignment operator

42. What is the use of printf() and scanf() function?

- **printf()** function is used to display the results on the standard output (screen)
- **scanf()** function is used to read a value from the keyboard (standard input),

43. What are the three types of character taken by the control string of printf() function?

i) Ordinary characters
ii) Formatting characters
iii) Escape sequence characters

44. List the formatting characters in C.

Formatting Character	Data type
%d	Int
%f	Float
%c	Char
%s	char[ ]
%ld	long int
%lf	long float or double

45. What is calling function?

The function which calls another function is termed as calling function and the other is termed as **called function.**

46. What is function call?

The function call specifies the function name and provides necessary information as parameters that the called function needs in order to perform its specific task.

47. What is the difference between function prototype and function header?

Function Prototype	Function Header
Function declaration statement is terminated by semicolon	Function header statement is not terminated with semicolon
It is placed above the main() function	It is the first statement of the function

48. Compare formal and actual parameters.

Actual Parameters	Formal Parameters
Actual Parameters are the parameters defined in the calling function.	**Formal Parameters** are the parameters defined in the called function.
They have the actual values to be passed to the called function	They receive the values of the actual parameters when the function is invoked

49. What is Call by Value?

In Call by value, values of the actual parameters are copied to the formal parameters. Changes to the copy in the called function do not affect the original variable's value in the calling function.

50. What is call by address or call by reference?

In Call by address, the called function knows the address of the original variable of the calling function and can modify the variable's value of the calling function.

51. What are the attributes of a variable?

- Name
- Type
- Value

52. List the storage classes provided by C.

- auto
- static
- register
- extern

53. List the conditional statements in C.

- if statement
- nested if-else structure
- switch case statement

54. Write the syntax of if statement.

```
if(relational expression)
      statement;
if(relational expression)
      statement1;
else
     statement2;
```

55. What is the use of break statement?

The **break** statement is used to end processing of a particular case statement within the **switch** statement.

56. What is loop? List its types.

A loop is a part of a program used to repeat a set of statements until certain specified conditions are met.

Types:

i) while
ii) for
iii) do while

57. What is the use of getchar() function?

getchar() is used to read a character at a time from the keyboard and it is a pre-defined function.

58. What is the difference between while and do-while loop?

while loop	do-while loop
Condition is tested at the **entry level**	Condition is tested at the **exit level**
Loop executes only if the condition is true	Loop executes at least once whether the condition is true or false

59. What is an array? What are the two types of array?

An array is a collection of homogeneous elements of similar data type.

Types:

1) Single dimensional array
2) Multi dimensional array

60. How do you access array elements?

The array elements can be accessed using **indices.** An array index starts from **zero** to **n-1**

61. What are the operations that can be carried out using pointer?

- An integer can be added to or subtracted from a pointer
- Two pointers can be subtracted

62. What is the purpose of strlen() function?

The function strlen() is used to find the length of the string. This function is available in **string.h** file.

Syntax: **strlen(char *);**

Example: **int l = strlen(name);**

****

THREE MARKS QUESTIONS AND ANSWERS

1. Give the important differences between the flow chart and the pseudo code.

FLOW CHART	PSEUDO CODE
Flows of computational paths are depicted as a picture	Represents a procedure for doing something
Standard symbols are used	No standard styles are used
Can be used for small problems	Can be used for big problems
Precise	Not precise
Difficult to convert into a high-level langua computer program	very easy to convert into a high-level language computer program

2. State three differences between definite and indefinite iterations.

IDEFINITE ITERATION	INDEFINITE ITERATION
Exactly know how many times the iteration to be performed	Does not know exactly how many times the iteration is to be performed
Iteration stops, if the Answer is NO	Iteration never stops
Count is going to work here	Count is not going to work here

3. Give the properties of an algorithm.

- There should be a finite number of steps.
- Each step is executable without any ambiguity.
- Each step is executable within a finite amount of time, using a finite amount of memory space.
- The entire program should be executed within a finite amount of time.

4. Write short notes on pseudo code.

- Pseudo code is an intermediary language
- It is between English and the high-level computer languages.
- Pseudo code can be used to represent a procedure for doing something
- It is easy to understand things written in pseudo code

5. Write pseudo code to the fundamental control structures for branching and iteration. Give examples.

Branching:

- If .... then .... else ....
- If .... then ....

Example:

- **If** a > b **then** print a else print b
- **If** a < 10 **then** b = c + d

Iteration:

- For ..... to .... do .....
- While .... do .....

Example:

- **For** i = 1 to 20 **do**
  n = n + i

- **While** sum < 100 **do**
  sum = sum + i
  i = i + 1

6. Write pseudo code to find the volume of a cone.

start
read length, breadth and height.
volume = length x breadth x height
print volume
end

7. Write pseudo code for the sum of 100 numbers.

start
sum = 0
n = 1
while n <= 100 then do
    read a
    sum = sum + a
    n = n + 1
print sum
end

8. Write pseudo code to find the prime number.

start
read n
for i = 1 to n-1 do
if i divides n then
    (write 'not a prime'
    exit program
)
write 'prime number'
end

9. What are the points to be noted while writing pseudo code?

- Within one 'if then else' statement, there is another 'if then else' statement. To show this clearly indentation is used.
- Only the inner statement is written with extra indentation.
- All the statements in a sequence have the same indentation.
- Just as we use brackets in Mathematics, here also we use brackets for bunching

10. Write short notes on C programming language.

- C language was developed by Dennis Ritchie at AT & T Bell Laboratories
- Originally it was a language used with UNIX operating system
- It is a general-purpose language.
- It is an efficient, flexible and portable language

11. What are tokens? (OR) List the basic types of elements in C

- The basic types of elements are collectively known as tokens.
- The C language is composed five basic types of elements. They are:
  - i) Constants
  - ii) Identifiers
  - iii) Operators
  - iv) Punctuation
  - v) Keywords

12. What is an escape sequence? Give examples.

- Character combination consisting of a backslash "\" followed by a letter is called **escape sequence**
- It is a non-printable character constant.

Examples:

'\a' - Bell (beep)
'\b' - Backspace
'\f' - Form feed
'\r' - Carriage return
'\n' - New line
'\0' - null character

13. What are the rules for naming a variable?

- The name of a variable can consist of alphabets (letters) and numbers.
- An underscore character can be used
- The variable name starts with an alphabet
- Its length may vary from one character to 32 characters.
- Number is not allowed as a first character in the variable name.

14. Give the reasons for the following invalid variable names.

Invalid variable name	Reason
123	The first character is a number
1abc	The first character is a number
x value	A blank character is used
x&y	& is not a valid character in a variable name
for	It is a keyword

15. What is pointer variable? How do you declare it?

- A pointer variable assumes only address as its value.
- Each variable takes some locations in the main memory according to its type.
- A pointer variable is declared as follows:
  Syntax: int *variable;
  Example: int *y;

16. Write short notes on address of(&) and indirection(*) operators.

address of(&)

To obtain the address of the variable, we have to use the "address of" operator (&).

Example: y = &x; // the **address of x** is stored into the **pointer variable y**.

indirection(*)

To retrieve the value of a variable through the pointer, we can use the "indirection" operator (*).

Example: x=10;
y = &x;

- **y** represents the address of the variable x
- ***y** represents the value of the variable x

17. What are logical operators?

Symbol	Type of operation	Associativity
&&	Logical AND	Left to right
||	Logical OR	
!	Logical NOT	

- Logical AND (&&) operator returns true if both of its operands evaluate to true.
- Logical OR (||) operator returns true when one or both of its operands evaluates to true.
- Logical NOT(!) operator returns an opposite result of its operand

18. What is the use of ternary operator?

Ternary operator is also called as **conditional operator**. The symbol used for this operator is ?: . It has three operands.

Syntax: conditional expression**?** expression 1 **:** expression 2**;**

Example: j = i < 0 ? - i : i;

- If the conditional expression is **true**, **expression 1** is evaluated.
- If the conditional expression is **false**, **expression 2** is evaluated.

19. List the punctuations and their uses.

Punctuation	Uses
[ ]	used to represent array index
{ }	used to cover the body of the function
( )	used to represent a function, to group items and to group expressions
< >	used to enclose a header file in a preprocessor statement
" "	used to represent string literals
' '	used to represent a character constant
// or /* */	used to represent a comment
;	used as a statement terminator
,	used to separate items

20. List the types of expression.

- Constant Expression: a=10;
- Variable Expression: a=b;
- Arithmetic Expression: c = a+b;
- Relational Expression: c = a > b;
- Assignment Expression: f = d = e;
- Postfix Expression: x=i++;
- Prefix Expression: x=++i;

21. Write short notes on function prototype? (OR) How do you declare a function?

A function declaration may be called as a **function prototype** or a **function model.** The function prototype has four components:

i) Name of the function
ii) Return value type
iii) Number of parameters
iv) Type of each parameter

Example: int add(int, int);

22. Write short notes on storage classes.

Storage class is another attribute that is associated with the variable. It is used to determine scope and lifetime of variables. Storage classes in C are:

1) auto
2) static
3) register
4) extern

23. Compare static and global variables.

	Static variable	Global variable
Creation	It is created only within a function	It is created above all functions i.e main()
Scope	It can be accessed and modified only within a function	It can be accessed and modified by all the functions
Life time	ends only when the entire program execution is over	ends only when the entire program execution is over

24. Write the syntax of switch case statement.

```
switch (conditional expression)
{
case constant-expression 1:
            ……
            break;
case constant-expression 2:
            ……..
            break;
.
.
default:
            ……..
}
```

25. How do you declare an array?

- An array declaration specifies the **name** of an array and the **type** of its elements.
- A constant(size) should be used within the square brackets that specify the number of elements in the array
- Size value must be greater than zero

Syntax:

data type arrrayname[size];

Example: **number of elements**

int a[100];

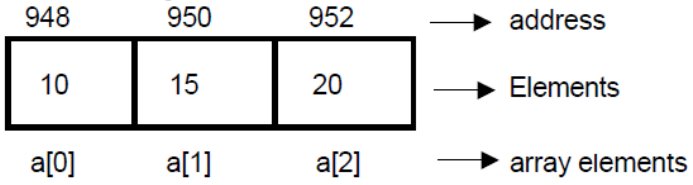
26. Write a C program segment to display whether the given character is vowel or consonant using switch case statement

```
char ch;
ch = 'a';
switch(ch)
{
   case 'a':
   case 'e':
   case 'i':
   case 'o':
   case 'u':
            printf("the given character is vowel");
            break;

    default:
            printf("the given character is consonant");
   }
```

27. Explain how the array elements are stored and represented in the memory.

- The elements of an array are stored in contiguous memory locations.
- The address of the first element is represented as &a[0].
- The compiler after allocating memory for the array, it stores the starting (base)address in the array name itself.
- both **a** and **&a[0]** represent the starting address
- indirection operator (unary *) is used to retrieve the value contained in a memory location



28. How do you read and print a string?

String is a collection of characters (an array) and its type will be char *.

i) To read a string from the keyboard:

scanf("%s", name);

- The formatting specification character %s is used to read a string. It cannot be used to read a string which contains blank spaces.

ii) To print the string:

printf("%s", name);

29. Compare array and structure in C.

Array	Structure
An array is a collection of elements of same data type	A structure is a collection of elements of different data types
An array is a homogeneous collection of elements	Structure is a heterogeneous collection of elements

30. How do you access the members of structure using pointers?

To access the members of the structure using the pointer, arrow operator (->) should be used instead of dot operator.

Example:

```
struct student *ptr;          /* declares structure pointer ptr */
struct student s1;
ptr = &s1;                    /* ptr points to the structure s1 */
ptr -> rollno;                /*  to access rollno field  */
```

31. Write short notes on array of structures.

- An array of structures can be declared as follows:

  struct student x[5];

- Here, **x** is an array of five structure elements.
- **x[0], x[1] ..., x[4]** are individual structure elements of type **struct student.**
- The members can be accessed as

  x[0].rollno, x[0].name, x[0].age

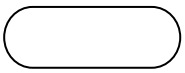
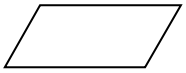
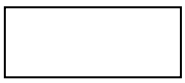
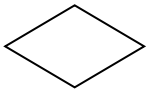
- To read an array of 5 student records, a for loop can be used

```
for(i=0;i<5;i++)
scanf("%d%s%d", &x[i].rollno, x[i].name, &x[i].age);
```

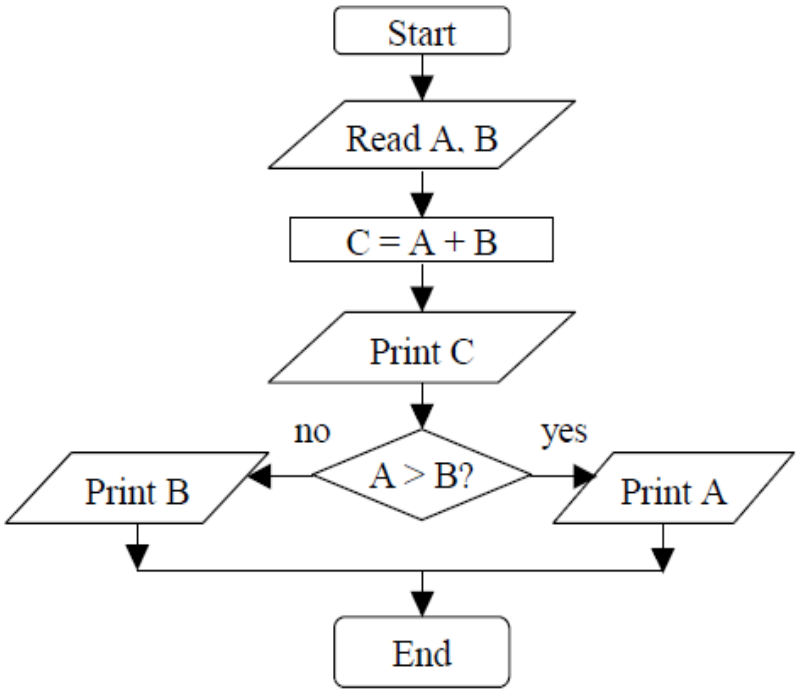

****

FIVE MARKS QUESTIONS AND ANSWERS

1. Draw the different types of boxes used in the flow chart. Explain each one of its roles.

SYMBOL	NAME	ROLE
	Oval	Start / End
	Parallelogram	Input / Output
	Rectangle	Calculations
	Diamond	Decision
	Arrow	Direction of control flow
	Circle	Connector

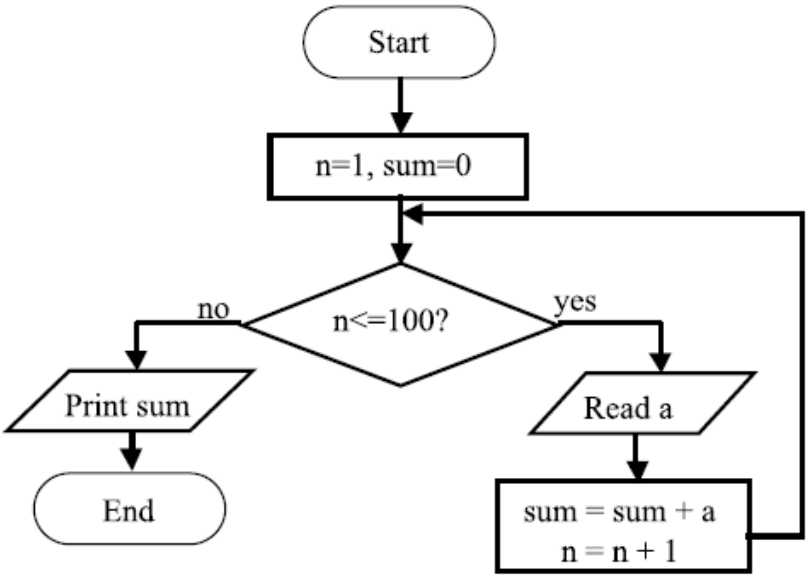
Example:



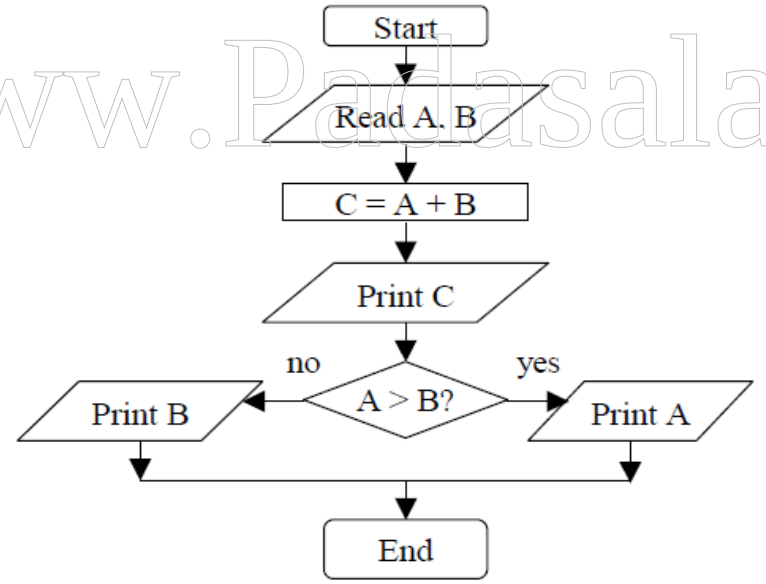
****

2. Give two examples of a two-way branching. Use a flow chart.

Example 1: Flowchart to read 100 numbers and prints their sum



Example 2: Flowchart to find the sum and maximum of two numbers



****

3. Give two examples of a multi-way branching. Use pseudo code

Example 1: Read a number between 0 and 3 and writes it in words

```
start
read n
(if n is 0 then print ‘zero’
1 then print ‘one’
2 then print ‘two’
3 then print ‘three’)
end
```

Example 2: Finding the minimum of 3 numbers.

```
start
read a, b, c
if a < b then
        (if a < c then
                print a
        else
                print c
        )
else
        (if b < c then
                print b
        else
                print c
        )
end
```

****

4. Explain the fundamental control structures using pseudo code.

There are three types of fundamental control structures. They are:

iii) Sequencing
iv)Branching
v) Iteration

i) Sequencing:

- Usually the calculations are done one after another, in a sequence.

Example: Finding the volume

```
start
read length, breadth and height.
volume = length x breadth x height
print volume
end
```

ii) Branching:

Two-way branching

- This is called the "If ...Then ...Else" structure.
- Ask a question. Get the answer as 'Yes' or 'No'.
- Depending on the answer, branch to one of the two available paths

Example: find the biggest of 2 numbers

```
start
read a, b
if a > b then print a
else print b
end
```

Multi-way branching

- For some questions, the answer can be one of many integers.
- Depending on the answer, we may go through different paths.
- This is called multi-way branching.

Example: Read a number from 1 -3 and write in word

start
read n
if n is 0 then print 'zero'
1 then print 'one'
2 then print 'two'
3 then print 'three'
end

iii) Iteration:

- repeating a set of actions again and again
- The action will be the same, but the data will change every time
- In this method we have to keep track of the count of the number of times the actions are performed.
- For this we use a variable called the **index variable** or **control variable**

Example: Sum of 20 numbers

start
sum = 0
for i= 1 to 20 do
sum = sum + i
print sum
end

****

5. Give two examples and illustrate the use of an index variable.

There are 4 basic steps involved in using an index variable.

i) The index variable should be given an integer as the initial value to start.

ii) The current value in the index variable **v** should be compared with the final value

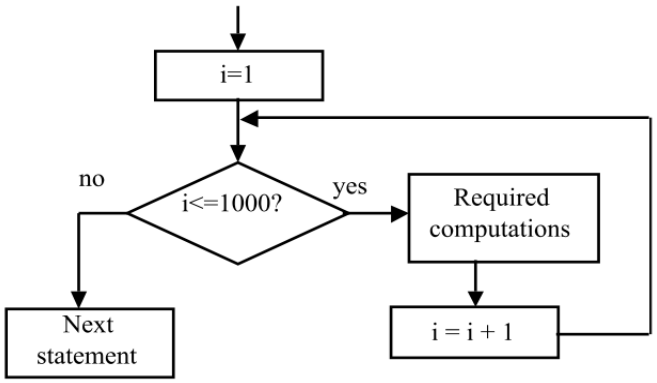
iii) If the answer is Yes, then

- Do the required actions once.
- Then increment the index **v** by 1.
- Go to step 2 and do the checking again.

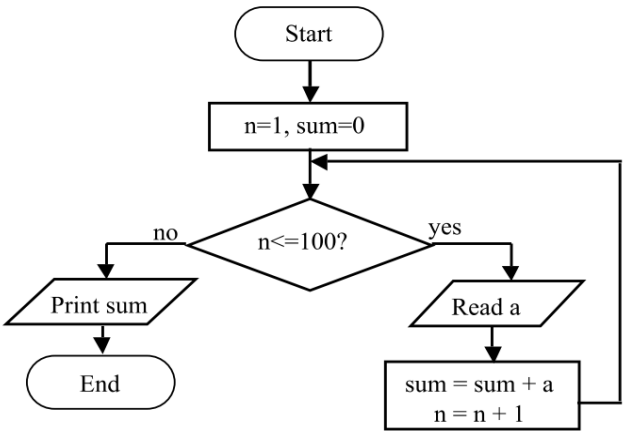
iv) If the answer is No, then

- The iterations are over.
- Go to the next action in the sequence.

Example 1: index variable is **i**



Example 2: index variable is **n**



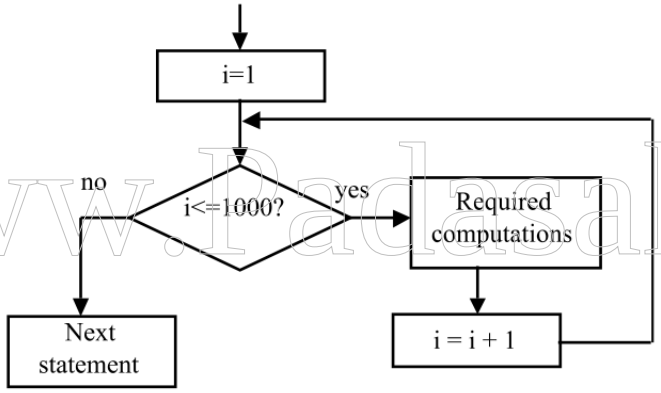
****

6. Explain definite iteration with an example.

Definite Iteration:

- Repeating a set of actions again and again is called as iteration.
- The action will be the same, but the data will change every time.
- Direct iteration is shown by the presence of loop
- A loop is formed by the directed lines.

Example:



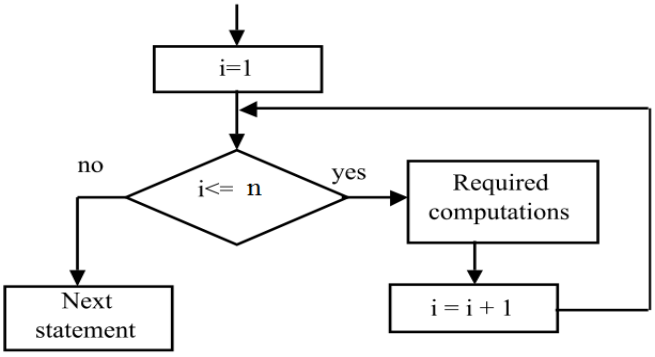
****

7. Using an example illustrate indefinite iteration.

Indefinite Iteration:

- In some situations, we may not know exactly how many times the iteration is to be performed
- Such iteration is called an indefinite iteration.

Example:



****

8. Write pseudo code to solve quadratic equation.

```
start
read a, b, c
if a = 0 then
(
    write 'this is not a quadratic equation'
    exit
)
else
(
    find d = b2 - 4ac
    if d < 0 then
            write 'imaginary roots'
    else
            (if d = 0 then
                    r = -b/a
                    write ' equal roots'
                    write r, r)
            else
                    (r1 = (-b + d)/ 2a
                    r2 = (-b - d)/2a
                    write 'unequal roots'
                    write r1, r2)
)
end
```


****

9. Explain constants in C programming.

Constant:

- A constant is of numeric or non-numeric type.
- It can be a number, a character or a character string that can be used as a value in a program.
- The value of a constant cannot be modified

Types of numeric constants:

i) integer constant
ii) floating-point constant
iii) character constant

i) <u>Integer constant</u>:

- An integer constant is a decimal number (base 10)
- It comprises of the digits 0 to 9.
- If an integer constant begins with the letters **0x** or **0X**, it is a **hexadecimal (base 16) constant**.
- If it begins with **0** then it is an **octal (base 8) constant.**
- Special characters are not allowed in an integer constant.

Examples: 23, 36, 0x1C, 0XAB, 071

ii) Floating - point constant:

- A floating-point constant is a signed real number.
- It includes integer portion, a decimal point, fractional portion and an exponent.
- An exponent is represented in powers of 10 in decimal system.
- The letter **E** or **e** is used to represent the floating-point constant in exponent form.

Examples: 58.64, 5.864×10^1 5.864E1

iii) Character Constant:

- A character is a letter, numeral or special symbol
- Single quotes are used to represent the character constant
- The characters used in C language are grouped into three classes.
  1) Alphabetic characters a, b, c, …., z, A, B, C, ….., Z
  2) Numeric characters 0 through 9
  3) Special characters + - * / % # = , . ‘ “ ( ) [ ] :

Examples: ‘1’, ‘a’, ‘+’, and ‘-‘

****

10. Explain the different types of binary operators used in C.

i) Arithmetic operators:

Symbol	Type of operation	Associativity
+	Addition	Left to right
-	Subtraction	
*	Multiplication	
/	Division (returns Quotient)	
%	Modulus (returns remainder)	

ii) Relational operators:

The relational operators are used to compare two values (items) and the result will be either true or false.

Symbol	Type of operation	Associativity
>	Greater than	Left to right
>=	Greater than or equal to	
<	Less than	
<=	Less than or equal to	
= =	Equal to (equality)	
!=	Not equal to (inequality)	

Logical operators:

The logical operators are used to connect two or more relational expressions.

Symbol	Type of operation	Associativity
&&	Logical AND	Left to right
||	Logical OR	
!	Logical NOT	

11. Explain Input and Output Statements with example. (OR) Explain printf() and scanf() functions.

i) Output statement:

- **printf()** function is used to display the results on the standard output (screen).
- The first parameter of the printf() function is a string which is used to control the output
- It is called as **"control string"** or **"formatting string"**

Syntax:

printf("formatting string", variables…)

Example:

```
int n;
n = 10;
printf("%d", n);
```

- Escape sequences allow partial control over the format of the output.

Example:

```
int i = 15;
printf("the value of i = %d \n", i);
```

The output is:
the value of i = 15

- The statement **printf("one\ntwo\nthree\n");**

The output is:

one
two
three

- The floating-point values are displayed with respect to six decimal places by default.

Example:

```
int x;
float y;
x = 10;
y = 10.5;
printf("%d %f", x, y);
```

The output is:
10 10.500000

ii) Input from keyboard

- To read a value from the keyboard (standard input), the function **scanf()** is used.
- The prototype of scanf() is similar to the prototype of printf().
- **address of(&)** operator is with a variable to provide the address of that variable.

Example:

```
int x;
scanf("%d", &x);
```

- While the scanf() function is being executed, the system waits for the user's input.
- The user has to provide data through keyboard.
- The data will be placed in the location of x only after "Enter" key is pressed in the keyboard.

12. How do you write a user-defined function? Explain.

A user-defined functions consists of :

i) Function Prototype or function declaration
ii) Function Definition
iii) Function Call
iv) Return Statement

i) Function Prototype or function declaration:

A function declaration may be called as a **function prototype** or a **function model**. It is terminated by semicolon. The function prototype has four components.

a. Name of the function
b. Return value type
c. Number of parameters
d. Type of each parameter

ii) Function Definition:

- Defining a function means to write a set of instructions (code) within curly braces { }.
- The code written within the curly braces is called as **function body** or a **block**.
- All variables declared in function definitions are **local variables.**
- The **function header** statement is the first statement in the function
- It is not terminated with semicolon.

iii) Function Call:

- Functions are invoked by a function call.
- The function call specifies the function name and provides necessary information as parameters

iv) Return Statement:

- **return** statement is used to return value from the function to the calling function.
- The function header has the **same data type** as the return value
- If the function does not return any value, then **void** keyword is used in function header

Example:

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int a, b,c;
        int add(int, int);              /* function prototype */
        a = 12;
        b = 11;
        c = add(a,b);                   /* function call */
        printf("%d\n", c);
}
int add(int x, int y)                   /* function header*/
{
return(x+y);                            /* return statement */
}
```

****

13. Explain how parameters are passed call by value to a function.

- When the parameters are passed call by value, a copy of the parameter's value is made and passed to the called function.
- Changes to the copy in the called function do not affect the original variable's value in the calling function

Example:

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int a, b,c;
        int add(int, int);              /* function prototype */
        a = 12;
        b = 11;
        c = add(a,b);                   /* function call by value */
        printf("%d\n", c);
}
int add(int x, int y)                   /* function */
{
return(x+y);
}
```

- When the assignment statement **c = add(a, b);** is being executed, the program control is transferred to the **add( )** function.
- When the **add( )** function is called, the values of the **actual parameters(a, b)** are copied to the **formal parameters(x, y)**
- When the function execution is over, the control is returned to the calling function where it is transferred

****

14. Explain how parameters are passed call by address to a function.

In Call by address, the called function knows the address of the local variable of the calling function and can modify the local variable's value of the calling function.

Example:

```
#include <stdio.h>
void main()
{
        int i;
        void change(int *);             /* function prototype */
        i = 20;
        change(&i);                     /* function call by reference */
        printf("%d\n", i);
}
void change(int *x)                     /* function */
{
*x = 23;
}
```

- The formal parameter **x** of the **change( )** function receives the address of the local variable **i** of
- the calling function
- Since **x** points to **i,** the value of **i** is modified.
- That is the value of the local variable of the calling function is changed.
- The program output should be **23**

****

15. Explain storages classes available in C.

Storage class is another attribute that is associated with the variable. C provides four storage classes:

i) auto
ii) static
iii) register
iv) extern

i) auto:

- **auto** variables are actually local variables.
- Their scope and lifetime are within that function
- They are created when the function is entered, and destroyed when the function is exited.
- We cannot access the values of the local variables outside the function

ii) static:

- If the variable has been declared as a **static**, its value will be retained even after the function execution is over.
- Scope of static variables are same as of local variables
- The life time of the static variable ends only when the entire program execution is over

iii) **register:**

- The **register** variables behave like auto variables.
- If a variable is declared as **register**, its value is placed in computer's registers.
- The register variables are used to speed up operations, by reducing memory access time.

iv) extern:

- Global variables are declared before the main() function.
- They can be **accessed** and **modified** by all the functions in the program.
- The **extern** variables have global scope
- The lifetime is throughout the execution of the program

Example:

```
#include <stdio.h>
int i = 0;                          /* extern variable */
void main()
{
        add( );
        register int a;             /* register variable */
        printf("%d",i);
}
add( )
{
        static int x;               /*static variable */
        int y;                      /* auto or local variable */
        i = i + 1;
}
```

16. Explain nested if statement with an example.

- The nested if-else structure is used to perform some operations based on choices
- If the first condition is true, only one comparison is made and all the other comparisons are skipped.
- When the first condition fails, the program continues to compare the second condition and it goes on similarly.
- This program works faster than **if-else-if** structure.

Example:

```
#include <stdio.h>
void main()
{
        int a,b,c;
        int choice;
        printf("Enter two integers: ");
        scanf("%d%d", &a,&b);                                   /* b is not zero */
        printf("1. addition\n");                        /* option 1 */
        printf("2. subtraction\n");                             /* option 2 */
        printf("3. multiplication\n");                          /* option 3 */
        printf("4. division\n");                        /* option 4 */
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if(choice == 1)
                c = a + b;
        else
        if(choice == 2)
                c = a - b;
        else
        if(choice == 3)
                c = a * b;
        else
        if(choice == 4)                                 /*comparison is optional */
                c = a / b;
        printf("the result = %d\n", c);
}
```


****

17. Explain switch case statement with an example.

- The switch – case statement is the modular replacement of nested if-else structure.
- The switch and case statements help to control complex conditional and branching operations.

Syntax:

```
switch (conditional expression)
{
case constant-expression 1:
                ……
                break;
case constant-expression 2:
                ……..
                break;
.
.
default:
                ……..
}
```

- The **switch(conditional expression)** and the **case constant-expression** must be **integer** type.
- Control passes to the statement whose **case value** matches with **conditional expression.**
- The **break** statement is used to end processing of a particular case statement within the switch statement.
- The **default** statement is executed if no case is equal to the value of conditional expression.
- The default statement is an optional

Example:

```
#include <stdio.h>
void main()
{
        int a,b,c;
        int choice;
        printf("Enter two integers: ");
        scanf("%d%d", &a,&b);
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                c = a + b;
                printf("%d", c);
                break;
            case 2:
                c = a - b;
                printf("%d", c);
                break;
            case 3:
                c = a * b;
                printf("%d", c);
                break;
            case 4:
                c = a / b;
                printf("%d", c);
                break;
            default:
                printf("the choice is out of range\n");
        }
}
```

****

18. Explain while loop statement with an example.

- The **while** statement is used to execute the set of statements repeatedly till the condition specified remains **TRUE**.
- In the while statement, the condition is tested at the **entry level.**
- The number of times the loop gets executed is controlled by a **control variable**
- The control variable is tested against a condition in the while statement
- It should be properly updated within the while loop for proper termination of the loop.
- If the updating line is missing, the value of the control variable will be always 1 and the loop never ends

Syntax:

```
Initialization of the control variable
while(condition)
{
      ……..;
                                  processing statements
      ……..;
       updating the control variable;
}
```

Example:

```
#include <stdio.h>
void main()
{
      int i;
      i = 1;                          /* Initialization */
      while(i <= 10)                   /*condition */
      {
            printf("%d\n", i);        /*processing statement */
            i = i + 1;                /*updating */
       }
 }
```

****

19. Explain for loop statement with an example.

- The **for loop** in C is simply a shorthand way of expressing a while statement
- **for** loop puts all three parts into one line.
- In the for loop, the condition is tested at the entry level
- The **control variable** is initialized first and then it is tested.
- If the test condition is TRUE, the body of the loop is executed; otherwise the loop is terminated

Syntax:

```
for(initialization; condition; updation)
{
body of the loop;
}
```

Example:

```
#include <stdio.h>
void main()
{
      int i;
      for(i=1;i<=10;i++)
      {
            printf("%d\n", i);

       }
 }
```

20. Explain do while statement with an example.

- In **do – while** statement, the condition is tested **at the exit level**
- So, the body of the loop is executed at least once whether the condition is true or false.
- At the end of the do – while loop, the condition is tested and if it is TRUE, the loop gets executed once again.
- When the test condition becomes FALSE, the loop is terminated nt

Syntax:

```
Initialization of the control variable
do
{
    statement;
    ……..
    updating the control variable;
} while(condition);
```

Example:

```
#include <stdio.h>
void main()
{
      int i;
      i = 1;
      do
      {
            printf("%d\n", i);
            i = i + 1;
       } while(i <= 10);
}
```


****

21. Explain single dimensional array with an example.

An array is a collection of homogeneous elements of similar data type.

i) Declaring an array:

An array declaration specifies the **name** of an array and the **type** of its elements. Size(index) value must be greater than zero.

Syntax:

```
data type  arrrayname[size];
```

Example:

```
int a[10];
```

ii) Array initialization:

```
int a[3]={10,15,20};
```

iii) Accessing array elements:

The array elements can be accessed using an index value. The index value starts from **0**.

iv) Assigning value for the array elements

The value for the array elements can be assigned as:

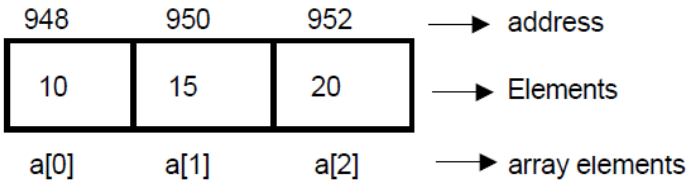
a[0]=10;

a[1]=15;

a[2]=20;

v) Storing array elements in the memory:

The elements of an array are stored in contiguous memory locations. The address of the first element is represented as &a[0].



Example:

```
#include <stdio.h>
void main()
{
        int a[10];
        for(i = 0;i<10;i++)
        {
                printf("Enter value for array elements\n");
                scanf("%d", &a[i]);
        }
}
```


****

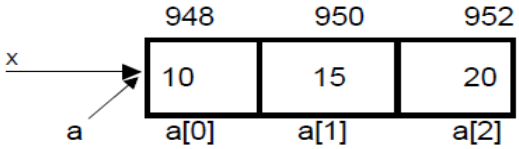
22. How arrays and pointers are closely related to each other? Explain.

- The **starting address** or the **base address** of an array is stored in the **array's name** itself.
- Since the address is stored in the array name it becomes a **pointer.**
- Consider the pointer variable **x**.

  int *x;
- Consider an array of 3 integers.

  int a[3] = {10, 15, 20};
- Here, **x** is a pointer variable which can assume an address of another integer
- **a** is a constant pointer to an integer, i.e., to the first element of the array.
- The **base address** of the array can be assigned to the pointer variable **x**.

  x = a;



- We can use the indirection operator * to retrieve the value contained in memory location.
- The expressions *(a+0) and *(x+0) both yield the value 10.

  ***(x + 0) < = > *(a + 0)**

  ***(x + 0) < = > x[0]**

****

23. Write any 2 versions of user-defined function to find the length of the string.

Version 1:

```
int lenstr(char *s)
{
        int count = 0;
        while(s[count] != '\0')
                count++;
        return(count);
}
```

Version 2:

```
int lenstr(char *s)
{
        int count = 0;
        while(*s != '\0')
        {
            count++;
             s++;
         }
         return(count);
}
```

Version 3:

```
int lenstr(char *s)
{
        char *start, *end;
        start=end=s;
        while(*end)
            end++;
        return(end-start);
}
```


****

24. Explain Multidimensional Arrays with an example.

A multidimensional array has been considered as an array of Arrays.

i) Declaration:

int a[3][3];

- The **first dimension** represents the **number of rows**
- The **second dimension** represents the **number of columns.**
- The array index starts from **0** in C language.
- We can access the first element using a[0][0].

ii) To read value of array elements:

Example:

```
int a[3[[3];
int i, j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
scanf("%d", &a[i][j]);
```

iii) To print the array elements in row wise:

```
int a[3[[3];
int i, j;
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
printf("%d", a[i][j]);
printf("\n");
}
```

****

25. Explain Structure in C with an example.

- Structures are derived data types in C language.
- Structure is used to create user-defined types.
- A structure is a heterogeneous collection of elements.
- Structures are commonly used to define records to be stored in files.

i) To Define a structure:

- The **struct** is a keyword, which is used to define a structure
- Variables (fields) declared within the braces of the structure definition are the structure's members
- The structure definition must end with a semicolon.
- The structure definition creates a new data type that is used to declare variables.

Example:

```
struct student
{
    int rollno;
    char name[24];
    int age;
};
```

ii) To create a structure variable:

Structure variables are declared like variables of other types.

Example:

```
struct student x, y;
```

- **x** and **y** are the variables of type struct **student**.
- Each variable has three fields as defined in the structure.
- A total of **28 bytes** will be allocated for each variable of type struct student.

iii) To declare structure variables while defining the structure:

Example:

```
struct student
{
    int rollno;
    char name[24];
    int age;
}x,y;
```

iv)Accessing the members of the structure:

- To access the members (fields) of a structure, dot operator is used.
- The structure variable is used as a qualifier along with the dot operator.

Example: to assign the roll numbers for the students **x** and **y**

```
x.rollno  = 1000;
y.rollno  = 1001;
```

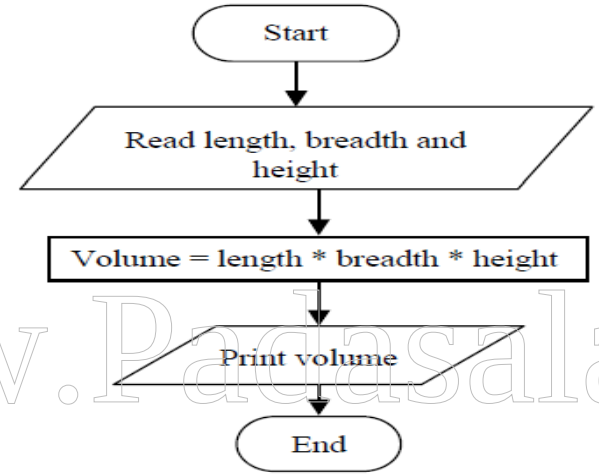
Example: To read the members of the student record, the function scanf() can be used

```
scanf("%d%s%d", &x.rollno, x.name,&x.age);
```

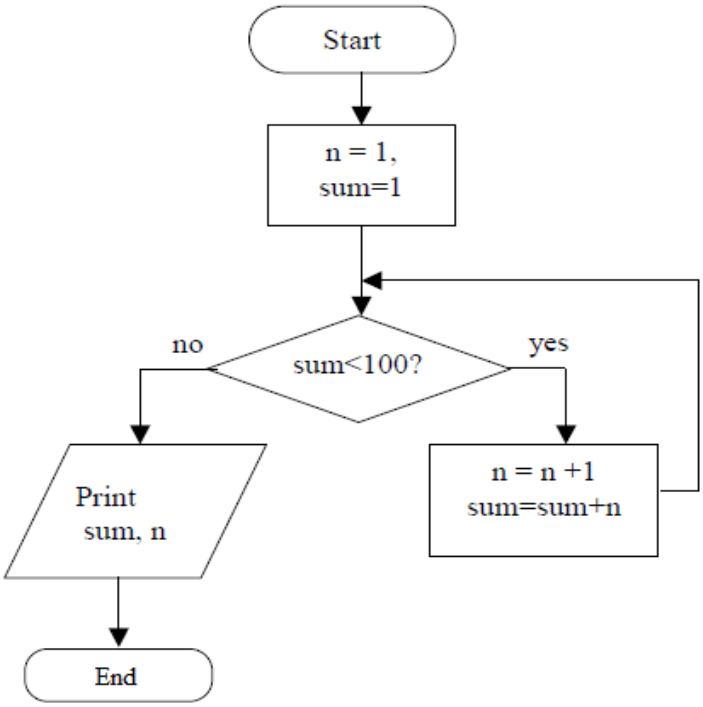
****

Flowchart Examples

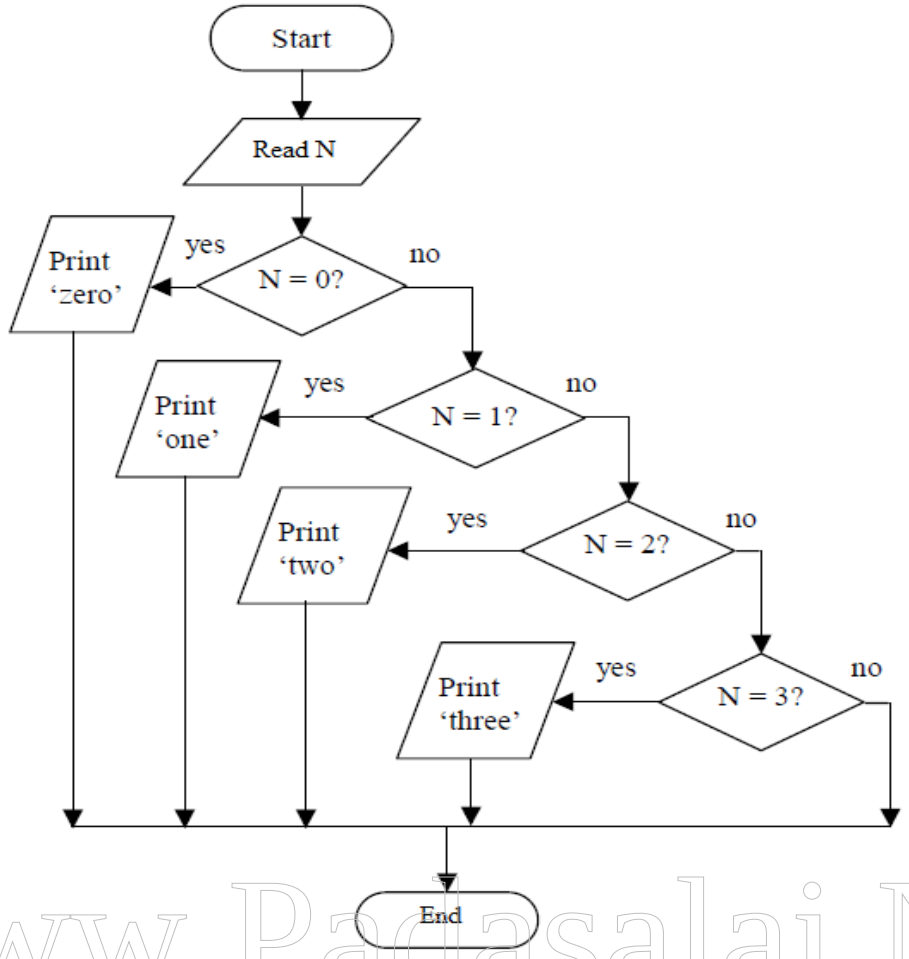
1. **Draw a flowchart to find the volume of a box using its length, breadth and height.**



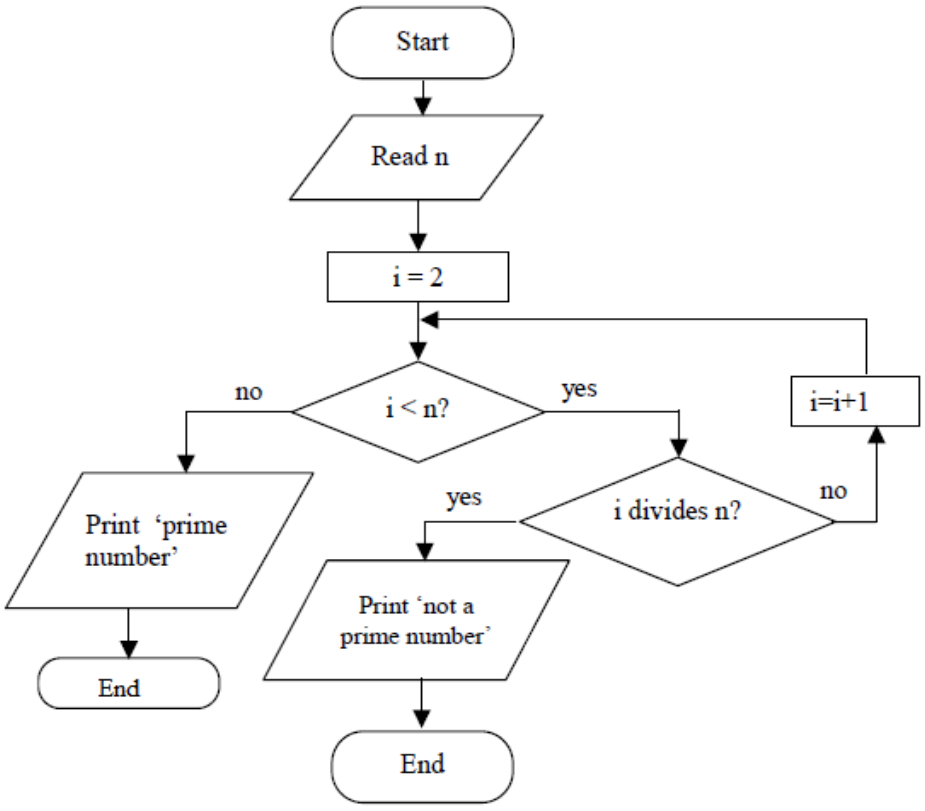
2. **Draw a flowchart to find the smallest integer n such that, 1+ 2+ 3+...+n is equal to or greater than 1000.**



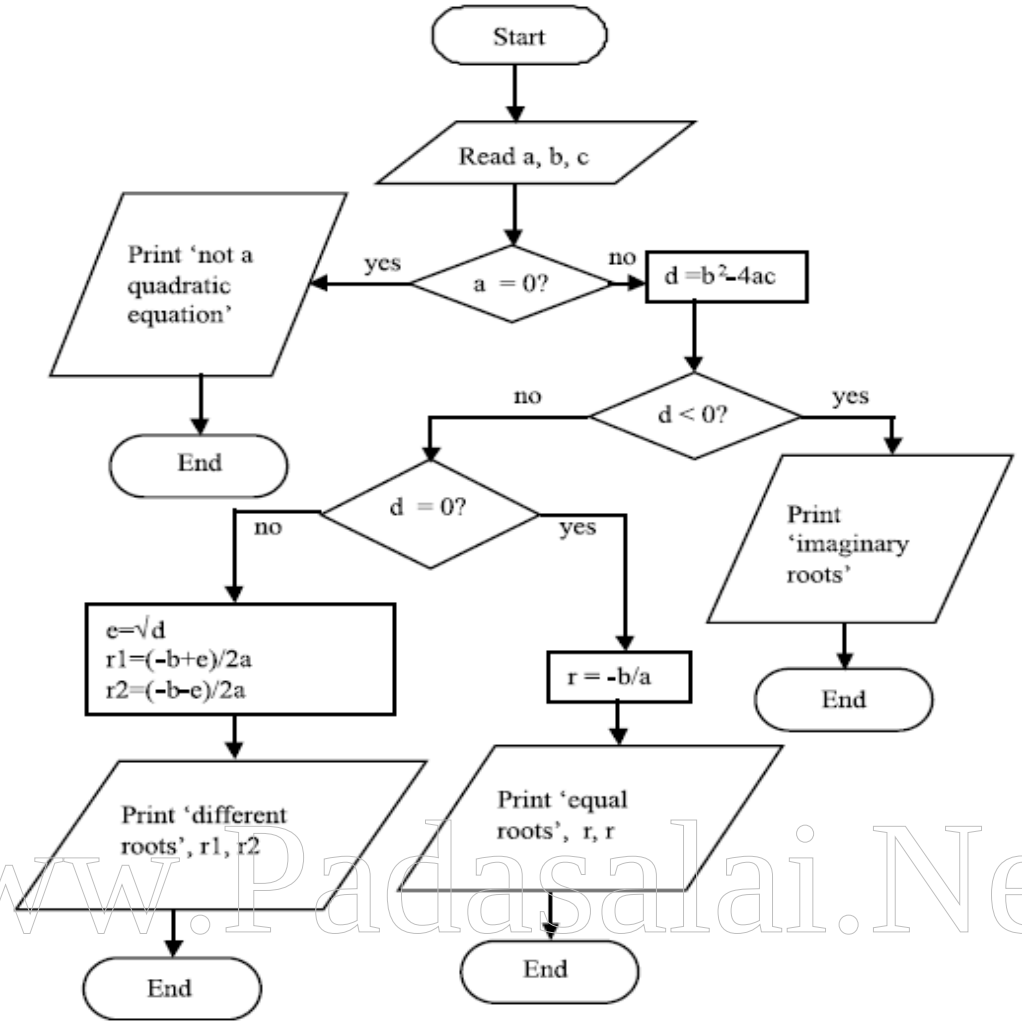
3. Draw a flowchart to read a number between 0 and 3 and writes it in words.



4. Draw a flowchart to determine whether a given integer is a prime number or not a prime a number.



5. Draw a flowchart to provide a method to solve the quadratic equation ax2 + bx + c = 0 .



****