C PROGRAMMING

NOTE:

We have designed the complete course material in accordance with new syllabus under WBSCTE. Short Answer Type and Long Answer Type Questions, those are relevant with new syllabus, are collected from Question Papers of old syllabus and included in the subject matter. Solution 2015 & 2017 question papers (new syllabus) are also included chapterwise.

Unit 1: Basics of C

Unit at a glance:

1.1 History of C

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972 By Dennis Ritchie.

Difference between structured programming language and object oriented language.

	Procedural/Structured programming	Object-oriented programming
1	The structure mainly consists of methods each performing a set of functions. Emphasis is given mostly on the methods	The structure mainly composed of classes, and methods within the classes through which the data in the class is accessed. Emphasis is given mostly on the data.
2.	The data inside the structure is less protected hence accessible by all methods inside and outside the structure.	The data inside the class is mostly private hence cannot be accessible directly outside the class body
3.	The structure is modularized but there is no relationship imposed on these modules. Accessing a module from another is mostly a method call.	The structure can be well organized by a establishing a relationship between several classes.
4.	As there is no strict relationship between several modules hence this type of programming is less scalable. Change in one module may affect the other and can result in major program changes.	As it is possible to establish a proper relationship between classes hence this type of programming is more scalable and is not susceptible to major programming changes.
5.	Needs more discipline while programming. The programming modularity and various methods linking can sometimes become very complex. This makes maintainability more difficult.	Due to the relationship model the segregation is much cleaner and hence maintainability is less tedious.

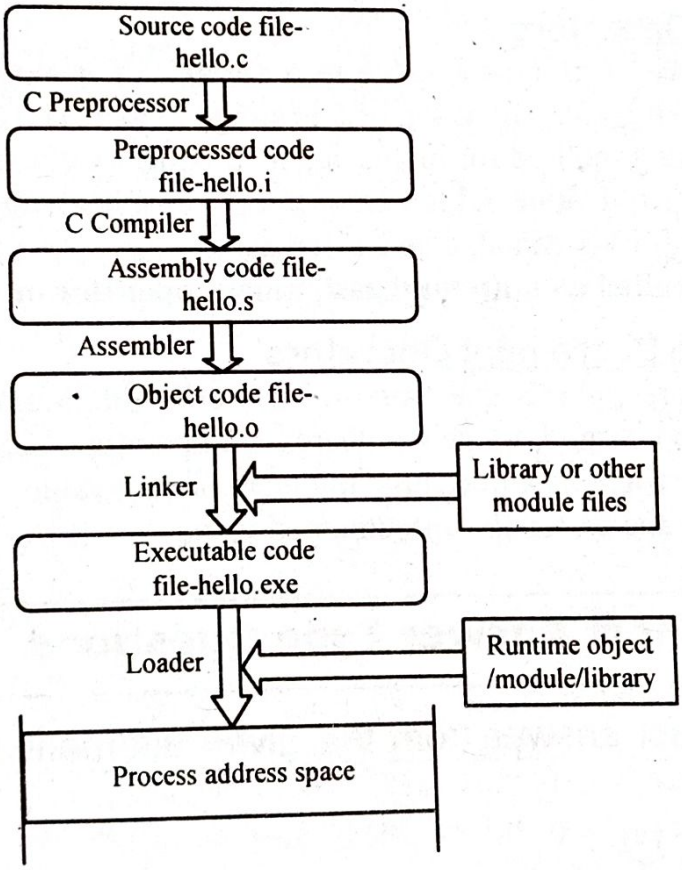
Files: A typical C program consists of the following

```
/*****This program is a sample program*******
*****This program demonstrates the useof different sections in C *****
*****This constitutes the comment line******/
#include<stdio.h>
/////this header file stdio.h contains all important library functions like print,scanf etc.This
has been included by macro definition of # include.
void main(){
int num;
///// this is the main function, return type is void meaning function doesnot return
anything to the compiler.
```

Printf("Enter number");///this constitutes a c statement asking from user, this to be printed on standard output device like monitor.
Scanf("%d", & num);///formatted input functions read from keyboard
Printf("You have entered %d\n", num);//formatted output functions
}
Save it as hello.c
Extension c denoted it is a c source file. When it is passed to c compiler, it reads the header files, expands it whenever necessary.
Normally C program building process involves four stages to get executable (.exe)

1. Preprocessing
2. Compilation
3. Assembly
4. Linking

The following Figure shows the steps involved in the process of building the C program starting from the preprocessing until the loading of the executable image into the memory for program running.



C character set,Tokens, Constants,Variables, Keywords

1. Any symbol that is used while writing C program is called alphabets or characters (Ex. A, b, c, +, *, i).
2. One or more alphabets can be grouped together to form meaningful words.These meaningful words are tokens.
3. One or more tokens can be grouped together to form meaningful sentences. These meaningful sentences are called statements or instructions.

4. One or more statements can be grouped together to form a complete program.
5. C supports several types of constants
   - Integer constants(e.g 139)
   - Real or Floating Point (Float) constants(e.g 24.89)
   - Character constants(e.g 'A')
6. A variable name is any combination of alphabets, digits or underscores of max size of 31 characters. (int i=2)

1.2 Various C Operators

1.2.1 Arithmetic Operators

The basic operators for performing arithmetic operations are:

- addition or unary plus (+)
- subtraction or unary minus (-)
- multiplication (*)
- division (/)
- modulus (remainder after division) (%)

1.2.2 Assignment Operators

The assignment operator = assigns a value to a variable. For example, x = 1, sets the value of x to 1, and a = b means the value of b is assigned to a. The expression **i = i + 1** is the standard programming method for increasing a variable's value by 1.
This expression takes i's old value, adds 1 to it, and stores it back into i.
The above expression i = i + 1 can also be written as:
i += 1 / / where += i s called a compound assignment operator or short hand operator .

1.2.3 Increment and Decrement Operators

C provides two operators for incrementing and decrementing variables. The increment operator ++ adds 1 to its operand, while the decrement operator – subtracts 1.
The prefix form ++i increments i and returns the incremented value. The postfix form i++ increments i, but returns the prior, non-incremented value.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. The escape sequence '\r' means –, **[WBSCTE 2006, 2007]**
   (a) new line (b) form feed (c) carriage return (d) tab

Answer: (c)

2. Write the output of the following program part: **[WBSCTE 2006]**

```
int q = 3959, d;
d=printf("%d", q);
printf("d=%d",d);
```

   (a) d = 395 (b) d = 0 (c) d =3 (d) d =4

Answer: (d)
(printf( ) returns the no. of characters printed. Here, 3959 is printed first, d is assigned by 4.)

3. int i = -1 ; then memory location of i contains – **[WBSCTE 2007, 2009]**
(a) FFFF (b) 8000H (c) -1 (d) none of these

Answer: (a)
(Since, 2's complement of -1 in Hexadecimal= FFFF)

4. The maximum negative value that a single integer variable can store is – **[WBSCTE 2007]**
(a) -64536 (b) -32768 (c) -32767 (d) -64545

Answer: (b)

5. If a = 11 and b = 3, what is the value of a%b? – **[WBSCTE 2007]**
(a) -3 (b) -2 (c) 3 (d) 2

Answer: (d)
(11 = 3×3+2. So, remainder is 2)

6. Which is the reserved word in C – **[WBSCTE 2007]**
(a) include (b) struct (c) null (d) main

Answer: (b)

7. Consider the variables char c = 'p'; int i = 6, f = 2; **[WBSCTE 2007]**
The result of the expression c<(i+f) –
(a) 0 (b) 1 (c) false (d) not defined

Answer: (a)
(Since ASCII value of 'p' is 112, which is not less than 6+2=8.)

8. char p = 67; printf("%c", p), then the output will be – **[WBSCTE 2007]**
(a) C (b) 67 (c) 2 (d) none of these

Answer: (a)
(67 is the ASCII value of C)

9. **[WBSCTE 2007]**

```
#define max(a, b) a<b?a:b
int a1 = 8, a2 = 9;
printf("%d", max( a1, a2));
```

The output will be –
(a) compilation error (b) run time error (c) 9 (d) 0

Answer: (c) (Since, 8<9? 8:9=9)

10. Suppose 'a' is an unsigned int variable and the value is 0x6DB7. The expression A>>6 will result in – **[WBSCTE 2007, 2008]**
(a) 0xCCB6 (b) 0x0DB7 (c) 0x01BC (d) 0x01B6

Answer: (d)
(On right shifting 0x6DB7 by 6 bits, we get 0x01B6)

11. typedef is a – [WBSCTE 2007]

(a) keyword, (b) predefined function

(c) both (a) and (b) (d) none of these

Answer: (a)

12. If a = -5 and b = -8 what is the value of a % b- [WBSCTE 2007]

(a) 0 (b) 1 (c) 3 (d) none of these

Answer: (d) (answer is -5)

13. '++' is a – [WBSCTE 2007]

(a) unary operator (b) binary operator

(c) ternary operator (d) can be used as both binary and unary operator

Answer: (a)

14. '&&' operator – [WBSCTE 2007]

(a) left associative (b) right associative

(c) can be used is left or right associative (d) none of these

Answer: (d)

(left to right associative.)

15. The precedence of bitwise AND operator '&' over bitwise OR operator '|' – [WBSCTE 2007]

(a) more (b) less

(c) not specified (d) none of these

Answer: (a)

16. What is result of 16 >> 2? – [WBSCTE 2007, 2010]

(a) 4 (b) 8 (c) 2 (d) 5

Answer: (a)

(16 → 0000000000010000>>2 = 0000000000000100)

17. What will be the value of c after execution of this program? [WBSCTE 2007]

```
void main ( )
   {int a = 9, b = 10, c;
   c=( b < a || b > a);
     printf("c=%d",c);
 }
```

(a) c = 1 (b) c = 0 (c) c = - 1 (d) none of these

Answer: (a)

(a=9, b=10. Therefore, b<a = 0(false) and b>a=1(true). So, c=(0 || 1) =1)

18. getch ( ); works like – [WBSCTE 2007]

(a) printf (b) scanf (c) variable (d) none of these

Answer: (d)

(getch( ) takes only one character at a time as input, but scanf( ) can take several data types as input at a time. getch( ) works like getchar( ) or getche( )).

19. Which is the reserved keyword in C? – [WBSCTE 2008, 2010]

(a) null (b) union (c) printf (d) string

Answer: (b)

20. If a = 15 and b = 4, what is the value of a%b? [WBSCTE 2008, 2010]
(a) -3 (b) 3 (c) 2 (d) none of these

Answer: (b)
($15=4\times3+3$. Therefore, remainder = 3)

21. Which of the following shows the correct hierarchy of arithmetic operations in C? [WBSCTE 2008]
(a) ( ), **, * or /, + or –
(b) ( ), **, *, /, +, -
(c) ( ), **, /, *, +, -
(d) ( ), / or *, - or +

Answer: (c)

22. The expression x = 4+2%-8 evaluates to – [WBSCTE 2008]
(a) -6 (b) 6 (c) 4 (d) none of these

Answer: (b)
(First, 2%-8 gives result 2, then 4+2 gives 6)

23. The library function exit( ) causes an exit from- [WBSCTE 2008]
(a) the loop in which it appears
(b) the block in which it appears
(c) the function in which it appears
(d) none of these

Answer: (d)

24. char a = 321; printf("%c", a), then the output will be – [WBSCTE 2010]
(a) 67 (b) 321 (c) B (d) A

Answer: (d)
(char data type can store from 0 to 255. So, 321 will be stored to a as 321-256=65, which is the ASCII value of 'A'.)

25. What is the value of a? : [WBSCTE 2010]

```
#define SQR(x) x*x
a = SQR(2+3);
```

(a) 25 (b) 11 (c) 13 (d) 10

Answer: (b)
(SQR(2+3) is equivalent to 2+3*2+3 = 11)

26. What is the output? [WBSCTE 2010]

```
int c, d = 90;
c = d--;
printf("%d %d", c, d);
```

(a) 89 90 (b) 90 89 (c) 89 89 (d) 90 90

Answer: (b)
(d-- stores 90 to c, and then its value is decreased)

27. What is the output of 64>>2? [WBSCTE 2010]

Answer: 16
(64 → 0000000001000000>>2 = 0000000000010000 → 16)

28. Pick the correct output: [WBSCTE 2010]

```
int p = 500, q = 5;
```

```
printf("%d", p ==q);
```

(a) 500 (b) 0 (c) 5 (d) 1

Answer: (b)

(Since, p is not equal to q, it gives false, i.e., zero value.)

29. What is the result of 16>>2? **[WBSCTE 2010]**

(a) 4 (b) 8 (c) 3 (d) 0

Answer: (a)

(16 → 0000000000010000>>2 = 0000000000000100)

30. int p = 35; q = 145; p = p ^ q; q = p ^ q; p = p ^q; printf("%d %d", p, q); then output will be - **[WBSCTE 2010]**

(a) 35 145 (b) 0 5 (c) 145 35 (d) none of these

Answer: (c)

(the operations – p=p^q; q=p^q; p=p^q; swaps the value of p and q.)

31. Consider the variable char c = 'p'; int i = 6, f = 2; the result of the expression c< (i+ f) is – **[WBSCTE 2010]**

(a) 0 (b) 1 (c) false (d) true

Answer: (a)

(The logical expression c<(i+f) is not true. In C, zero is treated as false value.)

32. (a) If a = 11 and b = 3 what is the value of a % b? – **[WBSCTE 2010]**

(a) –3 (b) –2 (c) 3 (d) 2

Answer: (d)

33. If x = y = z = –1 then what is the output of the following ++x||++y &&++z; printf ("z = %d", z) :- **[WBSCTE 2010]**

(a) 0 (b) –1 (c) 1 (d) 2

Answer: (b)

34. "int a = –2"; then memory location of a contains – **[WBSCTE 2010]**

(a) 8000H (b) –2 (c) FFFE (d) none of these

Answer: (d)

35. Write the output of the following program fragment main ( )

```
{
    main ( )
    printf ("Hello world");}
```

(a) Compilation error (b) stack overflow error **[WBSCTE 2010]**

(c) infinite loop (d) "Hello World"

Answer: (b)

36. A variable can have – **[WBSCTE 2010]**

(a) r-value only (b) l-value only

(c) r-value and address (d) none of these

Answer: (b)

37. Which bitwise operator is suitable for checking whether a particular bit is on or off? [WBSCTE 2011]

(a) && operator (b) & operator
(c) || operator (d) ! operator

Answer: (b)

38. In which order do the relational (i) arithmetic (2) logical (3) assignment (4) gets evaluated? [WBSCTE 2011]

(a) 2134 (b) 1234 (c) 4321 (d) 3214

Answer: (a)

39. Which of the following is the correct order of evaluation for the below expression? $z = x + y * z / 4\%2 - 1$ [WBSCTE 2011]

(a) $*/\%+-=$ (b) $=*/\%+-$
(c) $/*\%-+=$ (d) $*\%/-+=$

Answer: (a)

40. Which of the following correctly shows the hierarchy of arithmetic operations in C? [WBSCTE 2011]

(a) $/+*-$ (b) $*-/+$ (c) $+-/*$ (d) $/*+-$

Answer: (d)

41. Which of the following is the correct usage of conditional operators used in C? [WBSCTE 2011]

(a) $a > b?\ c = 30 : c = 40$ (b) $a > b?\ c = 30;$
(c) $\max = a > b?\ a > c?\ a : c : b > c?\ b : c$ (d) return $(a > b)?(a : b)$

Answer: (c)

42. Is the following statement a declaration or definition? [WBSCTE 2011]

extern int *i*;

(a) Declaration (b) Definition (c) Function (d) Error

Answer: (a)

43. Which of the following function is more appropriate for reading in a multi-word string? [WBSCTE 2011]

(a) printf(); (b) scanf(); (c) gets(); (d) puts();

Answer: (c)

44. By default a variable is assigned with _____ in static storage class [WBSCTE 2015]

(a) garbage (b) 1 (c) 0 (d) 10

Answer: (b)

45. The output of the following program segment will be:

```
void main()
{
 int x=4,y=3,z;
 z=x---y; printf("\n%d%d%d",x,y,z);
```

[WBSCTE 2015]

}

(a) 422 (b) 331 (c) 432 (d) 222

Answer: (b)

[WBSCTE 2017]

46. C is a

(a) Middle level language (b) High level language

(c) Low level language (d) none of these

Answer: (a)

[WBSCTE 2017]

47. If a is an integer variable, $a = 5/2$, will return a value

(a) 2.5 (b) 3

(c) 2 (d) 0

Answer: (c)

[WBSCTE 2017]

48. ASCII code for a – z ranges from

(a) 0 – 26 (b) 35 – 81 (c) 97 – 123 (d) none of these

Answer: (c)

[WBSCTE 2017]

49. From which standard stream does a C program read data?

(a) stdin (b) stdout (c) stderr (d) all of these

Answer: (a)

B. Fill in the blanks in the following statements:

50. By default ***int*** is the return type of a C function. [WBSCTE 2013]

51. The operator ***modulus*** can't be used with real operands. [WBSCTE 2013]

52. A variable declared inside a function by default assumes auto storage class. [WBSCTE 2013]

53. The ***break*** statement when executed in a switch statement causes immediate exit from the structure. [WBSCTE 2013]

54. The function strcat has ***two*** parameters. [WBSCTE 2013]

C. State whether the following statements are True or False:

55. void datatype in C language does not take any space to store it.

[WBSCTE 2006, 2007]

Answer: False

(void is a not a data type, it is a return type)

56. 3789L is a valid constant in C language. [WBSCTE 2007]

Answer: False (L does not have any meaning)

57. sizeof is a runtime operator. [WBSCTE 2007]

Answer: False (sizeof is a compile time operator)

58. An expression has an I-value. [WBSCTE 2007]
Answer: True

59. [ ] is not an operator in C language. [WBSCTE 2007, 2008]
Answer: True ([ ] is used to represent the index of an array)

60. Using #define preprocessor statement multi-line macro can also be defined in C language. [WBSCTE 2007]
Answer: True
(E.g.- #define PRINT printf("C")
#define P PRINT)

61. The variables declared as register type storage class are defined in CPU registers. [WBSCTE 2007]

Answer: True

62. 'static' storage class type variable can be used with compile time initialization. [WBSCTE 2007]

Answer: True

63. All static variables are automatically initialized to zero. [WBSCTE 2008]
Answer: True

64. The expression !(x<=y) is same as the expression x>y. [WBSCTE 2008, 2009]
Answer: True

65. All preprocessor directives begin with #. [WBSCTE 2008]
Answer: True (E.g. - #define, #include etc.)

66. The keyword void is a datatype in C. [WBSCTE 2009]
Answer: False (void is a return type in C)

67. An explicit cast can be used to change the expression. [WBSCTE 2009]
Answer: True (E.g. – int x = 4; float a = (float) x;)

68. The predicate!((x>=10)||(y==5)) is equivalent to (x<10)&&(y!=5). [WBSCTE 2009]
Answer: True
(!((x>=10)||(y==5)) → (!(x>=10))&&(!(y==5)) → (x<10)&&(y!=5))

69. In C, it is illegal to mix character data with numeric data in arithmetic operations [WBSCTE 2009]

Answer: False ('C'+4 is allowed in C)

70. Associativity is used to decide which of several different expressions is evaluated first. – [WBSCTE 2009]
Answer: False
(Precedence is used to decide which of several different expressions is evaluated first)

71. Assignment statements used with conditional operators must be enclosed within a pair of parenthesis. **[WBSCTE 2010]**
Answer: False (Parenthesis is not required.)

72. The value assigned to a real variable p by the statement p = 7/2 is 3.5. **[WBSCTE 2010]**
Answer: True (real variable means float type of data in C)

73. The negation '!' operator is unary. **[WBSCTE 2010]**
Answer: True. (E.g. - !4)

74. 'register' is a storage class in C. **[WBSCTE 2010]**
Answer: True

75. The code 'extern int i' will define a variable type integer. **[WBSCTE 2010]**
Answer: False ('extern int I' defines external variable of type integer)

76. The code 'static int i' will initialize 'i' to be zero. **[WBSCTE 2010]**
Answer: True (static variable automatically initializes to zero.)

77. '~' is used for 2's complementation – **[WBSCTE 2010]**
Answer: False
(It is used for 1's complement.)

78. '>>' is a Right Shift operator in C – **[WBSCTE 2010]**
Answer: True

79. The typecast operator is a unary operator. **[WBSCTE 2010]**
Answer: True

80. The bitwise operators can be used for floating point type operators. **[WBSCTE 2010]**
Answer: False
(Only integer type and character type data can be operated using bitwise operator.)

81. "typedef" is a reserve word in C-language. **[WBSCTE 2010]**
Answer: True

82. "&&" operator is left to right associative. **[WBSCTE 2010]**
Answer: True

83. printf( ) returns an integer value. **[WBSCTE 2010]**
Answer: True
(printf( ) returns the no. of characters printed.)

84. getch( ) function is used to get/read an integer data from a I/O file. **[WBSCTE 2010]**
Answer: False
(getch( ) is used to read a character from keyboard.)

85. 'p[a]' and '*(p+a)' are same with respect to r-value and l-value. [WBSCTE 2010]
Answer: True

86. "static" storage class type variable can be used with compile time initialization. [WBSCTE 2010]
Answer: True

87. 'switch' statement can be use to switch on variable of type 'char'. [WBSCTE 2013]
Answer: True

88. 'break' is used to take the control to the beginning in a 'switch' statement [WBSCTE 2013]

Answer: True

89. Assignment statements used with conditional operators must be enclosed within a pair of parenthesis. [WBSCTE 2013]
Answer: False

90. A 'dowhile' loop is used to ensure the execution at least once. [WBSCTE 2013]
Answer: True

91. The value assigned to a real variable 'p' by the statement p = 7/2 is 3.5 [WBSCTE 2013]

Answer: True

92. Too many recursive calls may result in stack overflow. [WBSCTE 2013]
Answer: True

93. All functions including 'main' in C an be called recursively. [WBSCTE 2013]
Answer: True

94. The code 'a++' will produce an error. [WBSCTE 2013]
Answer: False

95. The negation operator '!' is unary. [WBSCTE 2013]
Answer: True

96. The return type of a function is char by default . [WBSCTE 2013]
Answer: False

97. 'extern' is a Storage Class in C. [WBSCTE 2013]
Answer: True

98. The code 'static int i' will initialize 'i' to be 1. [WBSCTE 2013]
Answer: True

99. C uses 'column major representation' for storing multidimensional array in memory. [WBSCTE 2013]

Answer: False

100. The code 'int A [5] = {0}' will initialize A [4] to be a garbage value. **[WBSCTE 2013]**

Answer: True

D. Answer the following questions:

101. What would be the output of the following program? Justify your answer. **[WBSCTE 2007]**

```
main ( )
{int a, b, c, d, e, sum = 0 0, i;
Print (" enter Five numbers");
Scanf ("%d %d %d %d y.d", &a, &b, &c, &d, &e);
Sum = a + b + c + d + e;
for ( I = sum ; I < = 0; I - - )
if = (i = || i= = b || i = = c :: i = = d :: i = = e)
{printf ("%d",i);exit( );
}
}
```

Answer: Syntax error and logical error will be found

102. State the purpose of one Unary Operator used in C. **[WBSCTE 2008]**

Answer:

Unary operators work on one operand only. Increment (++) is a unary operator. ++ operator is used to increase the value of an integer or a character variable by unity.

103. What is the purpose of a complement operator? **[WBSCTE 2008]**

Answer: The complement operator makes bitwise one's complement.

104. Why does n++ execute faster than n=n+1? **[WBSCTE 2008, 2009]**

Answer:

The expression n++ requires two memory operations (read the value of n and write the updated value) to carry out the increment operation whereas, n=n+1 requires three memory operations (read the value of n, read the value of 1and write the updated value) to carry out this operation. That is why n++ is faster operation than n=n+1.

105. What is L-value?

Answer: **[WBSCTE 2008]**

L-value or location value is an address that is stored into.

106. Name the basic data types used in C.

Answer: **[WBSCTE 2008]**

Basic data types in C – char, int, float, double.

107. How do we round positive numbers?

Answer: **[WBSCTE 2008, 2010]**

If there is a number which is greater than or equal to 5, then the previous number is increased by one, otherwise remains unchanged E.g. 19.5 is rounded-off as 20; whereas 19.4 is rounded off as 19.

108. Write the output: [WBSCTE 2008]

```
main( )
{
    char ch;
    printf("%d", sizeof(ch));
}
```

Answer: 1 (Size of character type data is 1 byte).

109. What is an escape sequence? [WBSCTE 2008]

Answer:
The backslash character \ is called the escape character and is used to escape the usual meaning of the character that follows it. Some nonprinting and hard-to-print characters require an escape sequence. For example, the horizontal tab character is written as \t.

110. What will be the output of the statement: printf ("number = %.2f\n", 5.4391). [WBSCTE 2008]

Answer:
The output will be 5.44. ("%.2f" means, the floating point number 5.4391 will be printed after rounding off the number upto 2 decimal places.)

111. How can a variable width be specified in a scanf ( ) format string? [WBSCTE 2008]

Answer:
Variable width can be specified in a scanf ( ) format string by using appropriate format specifier. E.g. – scanf("%4d", 500); or scanf("%4.2f", 5.254785)

112. Which is more appropriate for reading in a multi-word string? gets( ), printf( ), scanf ( ), puts( ). [WBSCTE 2009]

Answer: gets( ).

113. What is the difference between getchar( ) and scanf( ) ? [WBSCTE 2009]

Answer:
getchar ( ) can take a single character only ; whereas, scanf ( ) can take several data types as input at a time.

```
char c;
c = getchar ( );
```

114. What will be the output of the following program? [WBSCTE 2009]

```
main ( )
{
    int x = 4, y , z;
    y = --x ;
    z = x--;
    printf("%d%d%d", x, y, z);
}
```

Answer:
Output will be: 2 3 3 (At y= --x, x becomes 3 and stored to y. At z= x--, x is stored to z and then its value is decreased to 2.)

115. Write the output of the following program fragment: **[WBSCTE 2009]**

```
main( )
{   int x = 4, y = 3, z;
    z = x- -y;
    printf("\n%d%d%d",x,y,z);
}
```

Answer: 331.

(--(decrement) operator has higher precedence than –(minus). So, the expression will be executed like (x--)-y)

116. Distinguish between the following pair:
automatic and static variables. **[WBSCTE 2009]**

Answer:

automatic → Variable declaration is varied by program.
static→ Variable declaration is fixed for a program.

117. What are the differences between unary and binary operator? **[WBSCTE 2009]**

Answer:

Differences between Unary and Binary Operator: Unary operator operates with the operand only; whereas, binary operator operates with two operands.

118. What is conditional operator? **[WBSCTE 2009]**

OR,

What do you mean by ternary operator? **[WBSCTE 2016]**

Answer:

Conditional Operator: Conditional Operator (? :) is ternary operator (demands 3 operands), and is used in certain situations, replacing if-else condition phrases. Conditional operator's shape is:

Condition_phrase ? phrase1 : phrase2;

If *conditional_phrase* is true, *phrase1* is executed and whole phrase is assigned with value of this phrase1. If the conditional phrase is false, then*phrase2* is executed, and whole phrase is assigned with phrase2 value.

119. Why C is called middle level language? **[WBSCTE 2009, 2010]**

Answer:

C is called a middle-level computer language as it combines the elements of high-level languages with the functionalism of assembly language. It gives or behaves as High Level Language through *Functions* - gives a modular programming and breakup. increased efficiency for reusability. Also it gives access to the low level memory through *Pointers*. Moreover it does support the Low Level programming i.e., Assembly Language.

120. What is an r-value? **[WBSCTE 2009]**

Answer: *r-Value:* An r-value is just a value -- 17, or 3.14 or something on that order.

121. What is the difference between pre increment and post increment operators? **[WBSCTE 2010]**

Answer:
Pre increment operator first increment and then execute. Post increment operator first executes and then increment.

122. What are static variables? **[WBSCTE 2010]**

Answer:
Static storage class is declared with the keyword static when the variable is defined. These variables are automatically initialized to zero upon memory allocation just as external variables are. Static storage class can be specified for automatic as well as external variables. Static automatic variables continue to exist even after the block in which they are defined terminates. Thus, the value of a static variable in a function is retained between repeated function calls to the same function. The scope of static automatic variables is identical to that of automatic variables. Static is a type of variable whose default initial value is 0, and the variable can be assigned once.

123. What do you mean by identifiers and keywords? **[WBSCTE 2010]**

Answer:
The keywords are reserved in C. Those keywords are restricted to use as variable or function name. C has 32 keywords. Identifiers are names given to program elements such as variables, arrays & functions. Basically identifiers are the sequences of alphabets or digits.

124. What is the difference in const and volatile? **[WBSCTE 2010]**

Answer:
Const is used with a datatype declaration or definition to specify an unchanging value. If we declare a variable as const then the value of that variable will not be changed. volatile specifies a variable whose value may be changed by processes outside the current program.

125. How symbolic constant is defined in C? **[WBSCTE 2010]**

Answer:
Using #define statement, symbolic constants are defined in C.

126. Is it possible to provide arguments to main( )? Explain. **[WBSCTE 2010]**

Answer:
Yes, it is possible to provide arguments to main( ). As main( ) is a function, we can pass parameters to the main( ). E.g.- float main(int a, int b).

127. What is a null statement? **[WBSCTE 2010, 2013]**

Answer:
A null statement consisting of only a semicolon and performs no operations.

```
int main()
{int i;
clrscr();
for(i=1;i<5;i++)
{if(i%2==0)
; <------------------------------- null statement
else
```

```
printf("%d\n",i);
}
getch();
return 0;
}
```

128. Assuming x = 10, evaluate the following logical expression: x == 10 || x>10 && !x **[WBSCTE 2010]**

Answer:

Non-zero value. (According to precedence, the expression will be executed like (x==10)||((x>10)&&(!x)).

129. Write the output of the following program fragment: **[WBSCTE 2010]**

```
main( )
{
    int x = 100, y =200;
    printf("%x", (x>y)?x:y);
}
```

Answer:

0x00C8. ((100>200)?100:200 gives 200. The hexadecimal equivalent gives 0x00C8)

130. What is the output? **[WBSCTE 2010]**

```
int a;
a = 11/3;
printf("%d", a);
```

Answer: 3. (11=3×3+2. So, the quotient is 3)

131. Explain the difference between declaration and definition. **[WBSCTE 2011]**

Answer:

In declaration we just specify the type and no memory is allocated to the variable.
But in the definition an initial value is assigned and memory is allocated to the variable.

132. What does the data type of a variable signify? **[WBSCTE 2011]**

Answer:

C has a concept of 'data types' which are used to define a variable before its use. The definition of a variable will assign storage for the variable and define the type of data that will be held in the location.

133. Why do we need signed and unsigned integer? **[WBSCTE 2011]**

Answer:

1. Signed integers can hold both positive and negative numbers.
2. Unsigned integers can hold a larger positive value, and no negative value.

134. Differentiate between typecasting and type conversion. **[WBSCTE 2011, 2012]**

Answer:

In 'typecast' you're only temporarily considering a variable as a different type.

Whereas in 'type conversion' you actually change the variable permanently(unless you change it again).

135. Why should we not use floating point numbers for equality in expressions? **[WBSCTE 2011]**

Answer:

First of all, floating point values are not "random" in their behavior. Exact comparison can and does make sense in plenty of real-world usages. But if you're going to use floating point you need to be aware of how it works. Erring on the side of assuming floating point works like real numbers will get you code that quickly breaks. Erring on the side of assuming floating point results have large random fuzz associated with them will get you code that appears to work at first but ends up having large-magnitude errors and broken corner cases.

136. Explain the difference between declaration and definition. **[WBSCTE 2011]**

Answer:

In declaration we just specify the type and no memory is allocated to the variable. But in the definition an initial value is assigned and memory is allocated to the variable.

137. Why do we need signed and unsigned integer? **[WBSCTE 2011]**

Answer:

1. Signed integers can hold both positive and negative numbers.
2. Unsigned integers can hold a larger positive value, and no negative value.

138. What is the difference between a statement and a block? **[WBSCTE 2011]**

Answer:

C statements consist of tokens, expressions, and other statements.

A block is a collection of individual statements.

Statement: x = ++i;

Block: ++i;

x = i;

139. How is a comment formed in C? **[WBSCTE 2011, 2012]**

Answer:

Comments in C begin with a slash followed by an asterisk. Any text may then appear including newlines. The comment is finished with an asterisk followed by a slash.

Example: /* This is a comment */

140. Can comments be nested? **[WBSCTE 2011]**

Answer: Not in standard (K&R) C.

141. What is type checking? **[WBSCTE 2011, 2012]**

Answer:

The process by which the C compiler ensures that functions and operators use data of the appropriate type(s). This form of check helps ensure the semantic correctness of the program.

142. What is the difference between an l-value and r-value? **[WBSCTE 2011]**

Answer:

The lvalue refers to the left-hand side of an assignment expression. It must always evaluate to a memory location. The rvalue represents the right-hand side of an assignment expression; it may have any meaningful combination of variables and constants.

143. Differentiate between typecasting and type conversion. **[WBSCTE 2011]**

Answer:

In 'typecast' you're only temporarily considering a variable as a different type. Whereas in 'type conversion' you actually change the variable permanently(unless you change it again).

144. What is the function of gets and puts? **[WBSCTE 2011]**

Answer:

gets → To input a strong form keyboard at a time

puts → To display a strong to monitor at a time.

145. What do you understand by the term preprocessor directive? **[WBSCTE 2011]**

Answer:

Preprocessor directives, such as #define and #ifdef, are typically used to make source programs easy to change and easy to compile in different execution environments. Directives in the source file tell the preprocessor to perform specific actions. For example, the preprocessor can replace tokens in the text, insert the contents of other files into the source file, or suppress compilation of part of the file by removing sections of text. Preprocessor lines are recognized and carried out before macro expansion. Therefore, if a macro expands into something that looks like a preprocessor command, that command is not recognized by the preprocessor.

Preprocessor statements use the same character set as source file statements, with the exception that escape sequences are not supported. The character set used in preprocessor statements is the same as the execution character set. The preprocessor also recognizes negative character values.

146. Distinguish between getch() and getche(). **[WBSCTE 2011]**

Answer:

getch ( ) - It just accepts a keystroke and never displays it and proceeds further. Normally we use it at the end of the main ( ).

getche ( ) - It will accept a character from keyboard, displays it immediately and does not wait for Enter key to be pressed for proceeding.

147. What is the difference between an 1value and an rvalue? **[WBSCTE 2012, 2013]**

Answer:

An *lvalue* (*locator value*) represents an object that occupies some identifiable location in memory (i.e. has an address). *rvalues* are defined by exclusion, by saying that every expression is either an *lvalue* or an *rvalue*. Therefore, from the above definition of *lvalue*, an *rvalue* is an expression that *does not* represent an object occupying some identifiable location in memory.

148. Explain the usefulness of default statement in switch case statement.

Answer: [WBSCTE 2012, 2013]

Default statement is used to execute all statements whi ch are required to be executed if any of the conditions do not match or even when other necessary handling required to be executed.

149. What does array name signify?

Answer: [WBSCTE 2012]

An array name is interpreted as the address of the starting location of the array.

150. What is the meaning of scope of a variable? [WBSCTE 2012, 2013]

Answer:

Scope of a variable is the lines of code from which it can be seen and/or manipulated. *Scope* can be thought of as visibility.

151. How symbolic constant is defined in C? [WBSCTE 2013]

Answer: #define MAX 10

152. What is the difference between an 1 value and r value? [WBSCTE 2013]

Answer:

Expressions that refer to memory locations are called "l-value" expressions. An l-value represents a storage region's "locator" value, or a "left" value, implying that it can appear on the left of the equal sign (=). L-values are often identifiers. The term "r-value" is sometimes used to describe the value of an expression and to distinguish it from an l-value. All l-values are r-values but not all r-values are l-values.

153. What do you mean by sizeof () operator? [WBSCTE 2015]

Answer:

The sizeof operator yields the size in bytes of the operand, which can be an expression or the parenthesized name of a type. It has the following syntax:

sizeof unary-expression

sizeof (type-name)

154. What is header file? [WBSCTE 2015]

Answer:

A header file contains information on functions that is not contained in the same program file. These functions are defined either in another program file or within library.

155. Define compiler directive statement. [WBSCTE 2015]

Answer:

The compiler directive statements /FREE... /END-FREE denote a free-form calculation specification block. The compiler directive statements /TITLE, /EJECT, /SPACE, /COPY, and /INCLUDE allow us to specify heading information for the compiler listing, to control the spacing of the compiler listing, and to insert records from other file members during a compile. The conditional compilation directive statements /DEFINE, /UNDEFINE, /IF, /ELSEIF, /ELSE, /ENDIF, and /EOF allow us to select or omit source

records. The compiler directive statements must precede any compile-time array or table records, translation records, and alternate collating sequence records (that is, ** records).

156. What do you mean by operator precedence? **[WBSCTE 2015]**

Answer:

Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator. For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

157. Define typecasting. **[WBSCTE 2015]**

Answer: Type casting is a way to convert a variable from one data type to another data type. For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'. You can convert the values from one type to another explicitly using the cast operator as follows –

(type_name) expression;

158. What is the difference between 'a' and "a"? **[WBSCTE 2015]**

Answer: 'a' is a character literal . "a" is string . It has type char * (it is an array of chars).

159. What do you mean by debugging? **[WBSCTE 2016]**

Answer:

Debugging is the routine process of locating and removing computer program bugs, errors or abnormalities, which is methodically handled by software programmers via debugging tools. Debugging checks, detects and corrects errors or bugs to allow proper program operation according to set specifications.

160. What do you mean by type defining in C? **[WBSCTE 2016]**

Answer:

Typedef is a keyword that is used to give a new symbolic name for the existing name in a C program. This is same like defining alias for the commands.

161. What do you mean by storage Class? **[WBSCTE 2016]**

Answer: ***Refer to Question No. 6 of Long Answer Type Questions.***

162. Define extern storage class. **[WBSCTE 2016]**

Answer: ***Refer to Question No. 6 of Long Answer Type Questions.***

163. Write down four binary operators of C. **[WBSCTE 2016]**

Answer:

```
+      Binary Plus Operator
-      Binary Minus Operator
==     Equal to Operator
<      Less than Operator
```

164. Distinguish between getche() and getch() function. **[WBSCTE 2016]**

Answer:

getch() : This function is used to accept single character from user, but it has no echo (display) on screen when it is pressed. Mostly it is used at the end of program, when user need to see final output of his/her program.

getche() : This function is used to accept single character from user, and it has echo (display) on screen when it is pressed.

165. Define L – value of an expression. **[WBSCTE 2016]**

Answer:

Expressions that refer to memory locations are called "l-value" expressions. An l-value represents a storage region's "locator" value, or a "left" value, implying that it can appear on the left of the equal sign (=). L-values are often identifiers.

166. What do you mean by scope and lifetime of a variable? **[WBSCTE 2017]**

Answer:

The **scope** of any variable is actually a subset of life time. A variable may be in the memory but may not be accessible, so the area of the program where we can actually access our variable is the scope of that variable.

The scope of any variable can be broadly categorized into three categories:

1) **Global scope**: When variable is defined outside all functions. It is then available to all the functions of the program and all the blocks program contains.

2) **Local scope**: When variable is defined inside a function or a block, then it is locally accessible within the block and hence it is a local variable.

3) **Function scope**: When variable is passed as formal arguments, it is said to have function scope.

- **Life time** of any variable is the time for which the particular variable outlives in memory during running of the program.

167. What will be the output of following code **[WBSCTE 2017]**

```
main()
{
      int x=3,y,z;
      y=x=10;
      z=x<10;
      printf("x=%d y=%d z=%d",x,y,z);
}
```

Answer: Output: x=10 y=10 z=0

Long Answer Type Questions

1. (a) What do you mean by compile time error, run time error and logical error? **[WBSCTE 2007, 2008]**

(b) Write the relation between header file and library. **[WBSCTE 2007]**

Answer:

a) ***Compile time error:***
Compile time error is also called syntax error. These errors are detected at the time of compilation. Theseerrors are those, involving violations of the syntax of a language. Fortunately, such errors are detected by the compiler when the statements are scanned during compilation process.

Run time error: These errors are occurred while running the program. These errors occur due to wrong input of variables.

Logical error: Logical errors are capable of doing much more harm than syntax errors. Logical errors usually permit the program to run and produce the wrong results. Such errors cannot be detected by the compilers, because they do not violate the rules of the language.

b) A header file contains information on functions that is not contained in the same program file. These functions are defined either in another program file or within library. A library is a collection of in-built functions to perform various operations. They are grouped together as a single unit for re-usability. If a programmer wishes to use a function contained in a library, they only need to specify the function name, following the function prototype. To call these functions, a programmer must have to include necessary header files at the beginning of the program.

➲ 2. How are the library functions usually packaged within a C compiler?
[WBSCTE 2008]

Answer:
A library is a collection of in-built functions, called library functions, to perform various operations. They are grouped together as a single unit for re-usability. The unit, which contains the packaged library functions, is called a header file. If a programmer wishes to use a library function, he/she only needs to specify the function name, following the function prototype. To call these functions, a programmer must have to include necessary header files at the beginning of the program. So, it is the header files, into which, the library functions are usually packaged within a C compiler.

➲ 3. What is an identifier? Give example
[WBSCTE 2008]

Answer:
Identifiers are names that are given to various program elements, such as variable, function and arrays.Identifiers consist of letters and digits.
Like: X, sum_5, _weather etc.
But 4th is not identifier cause identifier first character must a letter.

➲ 4. Why stdio.h is included in a C program?
[WBSCTE 2008]

Answer:
The header file stdio.h contains many I/O function prototype declarations that most C programs will reuse. The function prototype of printf and scanf functions for reading and writing are declared in this file. That is why stdio.h is included in a C program.

➲ 5. What are header files? How are they used in a C program?
[WBSCTE 2008, 2010, 2012, 2013]

Answer:

A header file contains information on functions that is not contained in the same program file. These functions are defined either in another program file or within library. Header files in C have .h extension. The #include is used in programs to use a header file. This include statement merely indicate that a function prototype exists. Any functions declared within the Header file, referenced in the source code, will need to be linked in order for the program to run. This is done by the C linker. In general, these header files contain constants and types along with prototypes for functions available in the library.

6. What are the different types of storage classes in C?

[WBSCTE 2008, 2009, 2010]

OR,

Give the features of each storage class with example. **[WBSCTE 2011, 2012, 2013]**

OR,

Define storage classes in C. Give a comparative study between each of them.

[WBSCTE 2015]

Answer:

Every C variable has a storage class and a scope. The storage class determines the part of memory where storage is allocated for an object and how long the storage allocation continues to exist. It also determines the scopes which specify the part of the program over which a variable name is visible, i.e. the variable is accessible by name. There are four types of storage classes available in C:-

(i) Automatic storage class,
(ii) Register storage class,
(iii) External storage class,
(iv) Static storage class.

(i) Automatic storage class: Automatic variables are declared at the start of the block. Memory is allocated automatically upon entry to a block and freed automatically upon exit from the block. The scope of automatic variables is local to the block in which they are declared, including any blocks nested within the block.

(ii) Register storage class: Register variables are a special case of automatic variables. Automatic variables are allocated storage in the memory of the computer; however, for most computers, accessing data in memory is considerably slower than processing in the CPU. These computers often have small amounts storage within the CPU itself where data can be stored and accessed quickly. These storage cells are called registers. Normally the compiler determines what data is to be stored in the register of the CPU at what times. However, the C language provides the storage class registers so that the programmer can "suggest" to the compiler that particular automatic variables should be allocated to CPU registers, if possible. Thus, register variables provide a certain control over efficiency of program execution.

(iii) External storage class: In some applications it may be useful to have data which is accessible from within any block and/or which remains in existence for the entire execution of the program. Such variables are called global variables, and the C language provides storage classes which meet these requirements. External variables

may be declared outside any function block in a source code file the same way any other variable is declared; specifying its type and name.

(iv) ***Static storage class:*** Static storage class provides a lifetime over the entire program, however; it provides a way to limit the scope of such variables. Static storage class is declared with the keyword static when the variable is defined. These variables are automatically initialized to zero upon memory allocation.

7. Differentiate between key word and reserve word. **[WBSCTE 2009]**

Answer:

Keywords: Keywords are the words whose meaning already been explained to the C compiler. The keywords cannot be used as variable names because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer.

Reserved words: The reserved words have standard, predefined meaning in C. These keywords can be used only for their intended purpose; they cannot be used as programmer defined identifiers.

8. What is the purpose of an external variable? What is its scope? **[WBSCTE 2009]**

Answer:

Using extern is only of relevance when the program you're building consists of multiple source files linked together, where some of the variables defined, for example, in source file file1.c need to be referenced in other source files, such as file2.c.

Extern variables are of global variables and can be declared in another source file which could be external/outside the current program scope.

9. (a) Explain how bitwise XOR operates. **[WBSCTE 2010]**

(b) Differentiate = and = = operator in C. **[WBSCTE 2010]**

(c) Explain the operation of && and || with example. **[WBSCTE 2010]**

Answer:

a) Bitwise XOR operator is a binary operator, and the sign of XOR operator is ^. Its associativity is left to right.

X	OPERATOR	Y	OUTPUT
0	^	0	0
0	^	1	1
1	^	0	1
1	^	1	0

Since the integer occupies 16 bits (2 bytes) in memory, the binary equivalent of x = 10 and y = 6 is –

x = 10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
y = 6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
x^y	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Therefore x^y = 12.

b) Difference between = and = = operator:

= operator	= = operator
(i) It is an assignment operator. (ii) This operator is used to assign a value into a variable. E.g. -, x=y The value of y is assigned into x.	(i) It is a relational operator. (ii) This operator is used to check relationship. Always the value of this type of statement is 0 or 1.

c) '&&' is a logical binary operator. '&&' is logical AND operator. Integer is only its operand type. It returns true, if both operands are true, otherwise false.

For example, let us consider the following expression:

a>b && x = = 10

The expression on the left is a >b and that on the right is x= = 10. The whole expression evaluates to true or 1, only if both expressions are satisfied, i.e., if a is greater than b as well as the value of x is equal to 10.

Logical OR

For example, let us consider the following expression:

a > b || x = = 10

The whole expression evaluates to true or 1, only if any one of the expressions are true i.e. if a is greater than b or the value of x is equal to 10, or if both of them are true.

10. What is a symbolic constant? How is it defined? **[WBSCTE 2010]**

Answer:

Symbolic constants are constants represented by symbols.... constants are values that do not change throughout the program execution.

e.g

```
#include<stdio.h>
#include<conio.h>
#define NUM 15
void main()
{
int a=NUM;
printf("The constant is %d",a);
getch();
}
```

here #define is a preprocessor. Its job is to replace all the entries named NUM as 15. So the compiler works with the constant 15.

11. What do you mean by Associativity of an operand? **[WBSCTE 2010, 2017]**

Answer:

Associativity: Associativity rule decides the order in which multiple occurrences of the same level operator are applied. Following things are described the summary of C operator with their associativity.

Operator	Associativity
( )	Left to right
[ ]	Left to right
+(unary)	Right to left
-(unary)	Right to left
++	Right to left

--	Right to left
!	Right to left
~	Right to left
*	Right to left
&	Right to left

12. Differentiate between – Bitwise and logical operators. **[WBSCTE 2010]**

Answer:

A logical operator interprets its operands as either true or false and always evaluate to a 1-bit true or false result. A bitwise operator interprets its operands as vectors and returns a vector.

For a bitwise operator, the logical operation is performed on each bit position of the operand(s). If one operand is shorter than the other, it is bit extended with zeros in the left-most bits to match the length of the longer operand. The result returned is a vector of equal length to the longest operand.

13. (a) Explain the steps of execution of C program?
(b) Explain run time error. **[WBSCTE 2010]**

Answer:

a) Steps of execution of C program:

(i) A C program is first saved using F2 button. A C program is saved in a .c extension file.

(ii) The program file is complied by pressing ALT+F9.

(iii) On getting compilation success result, the program can be run using CTRL+F9.

b) Run time errors are occurred while running the program. These errors often occur due to the internal limitations of a program. E.g.- A program that accepts data from the user and store it in an array. The array size is predefined. But, if it shows no available space in the memory, then it is not a fault of the user.

14. What is a null statement? How can it be useful in our programs? Define a function. Why are they needed? **[WBSCTE 2011, 2012]**

Answer:

A "null statement" is a statement containing only a semicolon; it can appear wherever a statement is expected. Nothing happens when a null statement is executed.

The correct way to code a null statement is:

Statements such as do, for, if, and while require that an executable statement appear as the statement body. The null statement satisfies the syntax requirement in cases that do not need a substantive statement body.

This example illustrates the null statement:

```
for ( i = 0; i < 10; line[i++] = 0 )
```

15. What are the arithmetic operators that are permitted on pointers? **[WBSCTE 2012, 2013]**

Answer:

* (value at operator), & (address at operator), ->(arrow),++(post increment), --(post decrement), +(addition),-(substraction).

Unit 2: Decision Control

Unit at a glance:

2.1 Introduction

In C language there are two types of decision-making

- Branching: deciding what actions to take
- Looping: deciding how many times to take a certain action

2.2 Control Flow

The flow of control of a language specifies the order in which computations are performed. Like most programming language C accomplishes the "flow of control" by the following statement and conditional loops.

- if-else conditional statement block
- while loop
- for loop
- do-while loop
- switch-case statement block
- break and continue statements

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. The 'continue' statement is used to continue – **[WBSCTE 2007]**
   (a) to the next statement of the current function
   (b) to the next statement of the calling function
   (c) to the begin of the current statement
   (d) none of these

Answer: (d) (to the begin of the current loop)

2. What will be the output of the following program? **[WBSCTE 2007]**

```
void main ( )
{  char x = 'd' ;
switch (x)
{   case 'b' : puts ("01 001");
break;
default: puts ("1 2 3");
break
case 'R' : puts ("I  II  III" );
}
```

```
}
```

(a) 1 2 3 (b) 01 001 (c) I II III (d) none of these

Answer: (a)
(Since, x='d' is passed as parameter in switch block, and any case does not match with x, so default block is executed.)

3. The escape sequence '\n' means – [WBSCTE 2007]
(a) new line (b) variable tab (c) vertical tab (d) horizontal tab

Answer: (a)

4. What is the output?: [WBSCTE 2010]

```
void main( ){
    int a=1, b=0;
    if(a, b){
    printf("EXAM");
    }
}
```

(a) exam is printed (b) EXAM is printed
(c) compiler error (d) nothing is printed

Answer: (d) (Since, if(1,0) will be treated as if(0))

5. The keyword used to transfer control from a function back to the calling function is – [WBSCTE 2011]
(a) switch (b) goto (c) go back (d) return

Answer: (d)

6. Write the output of the following program segment:

```
void main()
{int i=1,j=1;
for(;;)  {
if(i>5) break; else j+=i; printf("%d",j); i+=j;
                }}
```

[WBSCTE 2015]
(a) 23 (b) 32 (c) 33 (d) compilation error

Answer: Output will be '2 5'

7. Write down the output of the following program segment(s):

```
void main() {
    int i;
      for(i=0;i<=10;i++); printf("%d",i);
    }
```

[WBSCTE 2015]
(a) display 1 to 10 (b) 10 (c) 11 (d) compilation error

Answer: (c)

8. A do while loop is useful when we want that the statements within the loop must be executed [WBSCTE 2017]
(a) only once (b) at least once
(c) more than once (d) none of these above

Answer: (b)

B. Fill in the blanks in the following statements:

9. The ***break*** statement is used to skip a part of the statements in a loop. [WBSCTE 2009]

[Note: continue, or goto can also be the answer.]

10. The ***break*** statement when executed in a switch statement causes immediate exit from the structure. [WBSCTE 2010]

C. State whether the following statements are True or False:

11. A program stops its execution when a break statement is encountered. [WBSCTE 2008]

Answer: False (program stops its execution on getting exit( ) function)

12. A switch statement can always be replaced by a series of if.else statement. [WBSCTE 2009]

Answer: False
(switch statement cannot be used when complicated conditional expressions are used).

13: The initialization, test condition and increment parts may be missing in a for statement – [WBSCTE 2009]

Answer: True (Only two semicolon is must in a for statement.)

14. 'switch' statement can be used to switch on variable of type 'char'. [WBSCTE 2010]

Answer: True

15. 'continue' is used to take the control to the beginning in a 'switch' statement. [WBSCTE 2010]

Answer: True

16. A 'while' loop is used to ensure the execution at least once. [WBSCTE 2010, 2014]

Answer: False (do-while loop is used to ensure the execution at least once)

17. The code 'int A [5] = {0}' will initialize A [4] to be a garbage value. [WBSCTE 2013]

Answer: True

D. Answer the following questions:

18. **Write the output for the following program segments.** [WBSCTE 2008]

```
int main ( )
{
int i;
for (i=5; ++i; i-= 3)
printf("%d", i);
return 0;
}
```

Answer:

Output will be – 642

(Initially i=5. At the time of condition checking, i increases by 1, and the decrement statement decreases the value of i by 3. For 1st iteration, at condition check, i becomes 6, which is true value, and hence 6 is printed. For 2nd iteration, decrement statement decreases i to 3 and after that at condition checking, i becomes 4(true), and is printed. For 3rd iteration, i decreases to 1, and then at condition checking, i becomes 2(true value) and hence printed. Now, i decreases to -1 and for the next condition checking it increases to 0, which is false value, and hence loop terminates.)

19. Write the output for the following program segments: **[WBSCTE 2008]**

```
int main ( )
{
for (; printf("C";));
return 0;
}
```

Answer:

It will give compilation error. The proper for statement is

`for (; printf ("c");); - This will print "C" infinitely.

20. Write the output for the following program segments: **[WBSCTE 2008]**

```
int main ( )
{
for (i=5; --i;)
printf("%d", i);
return 0;
}
```

Answer:

The output will be – 4 3 2 1

(At condition checking statement, i is first decreased by 1 and then condition is checked whether the value is zero or not. The loop continues its iteration, until it gets zero at the condition checking statement.)

21. What is the utility of Break statement? **[WBSCTE 2008, 2010, 2017]**

Answer:

Break statement provides immediate exit from for, while, do, switch.

22. Will the following two programs give the same output? **[WBSCTE 2008]**

(a)
```
main( )
{
int digit = 0;
while(digit<=9) printf("%d\n", ++digit);
}
```

(b)
```
main( )
{
int digit = 0;
while(digit<=9) printf("%d\n", digit++);
}
```

Answer:

The output of program (a)	The output of program (b)
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9

So, the output of the programs (a) and (b) are not same.

23. Write the output of the following program fragment: **[WBSCTE 2009]**

```
main( )
{
int i=1, j=1;
for( ; ; )
{
if(i>5)
break;
else
j+=i;
printf("\n%d", j);
i+j;
}
}
```

Answer:

The output is →

2

4

.....

The loop will continue infinitely.

24. Find the error, if an, of the following looping segments. Assume that all the variables have been declared and assigned values. **[WBSCTE 2009]**

```
while( count!=10)
{
count =1;
sum=sum+x;
count=count+1;
}
```

Answer:

The loop runs infinitely.

25. State the difference between while and for loop. **[WBSCTE 2009]**

Answer:
The initialization, test-condition and increment is part of the for loop; but not in while loop; the test condition is the only part of while loop.

26. Find errors, if any, in the following looping segment: **[WBSCTE 2009]**

```
while (count:=10);
{
counts 1;
count=count+1;
}
```

Answer:
Syntax error will occur. At the condition checking statement of while loop, count:=10 has no meaning.

27. What are nested loops? **[WBSCTE 2010]**

Answer:
Nested Loop: A loop inside another loop is called nested loop.

28. What is the difference between while and do-while loop? **[WBSCTE 2010]**

Answer:
In while loop, at first condition checking is done then execution will occur. But in do while, at first loop block is executed then condition checking is done.

29. Do you know nested loop. Explain by an example. **[WBSCTE 2011, 2012]**

Answer:
A loop inside another loop is known as nested loop. We can write any loop inside any loop in c i.e. we can write for loop inside the loop or while loop or do while loop etc.
For example:

```
void main()
{
int i,j,k;
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
printf(" %d",i+j);
}
}
getch();
}
```

30. Explain the usefulness of default statement in switch case statement. **[WBSCTE 2011]**

Answer:
The default statement can occur in any order in the switch statement. The default clause is an optional clause that is matched if none of the constants in the case statements can be matched.

31. Compare the use of if-else construct with that of ternary operator. **[WBSCTE 2011]**

Answer:

The if-else construct (sometimes called if–then–else) is common across many programming languages. Although the syntax varies quite a bit from language to language, the basic structure (in pseudocode form) looks like this

```
IF (boolean condition)
  (consequent)
ELSE
  (alternative)
END IF
```

n C and C-like languages conditional expressions take the form of a ternary operator called the conditional expression operator, ?:, which follows this template:

(condition)?(evaluate if condition was true):(evaluate if condition was false)

This means that conditions can be inlined into expressions, unlike with if statements, as shown here using C syntax:

```
my_variable = if(x > 10) { "foo" } else { "bar" };
my_variable = (x > 10)?"foo":"bar";
```

To accomplish the same as the second (correct) line above, using a standard if/else statement, this would take more than one line of code :

```
if (x > 10)
  my_variable = 'foo';
else
  my_variable = 'bar';
```

The explicit if-else statement is easier to read and usually compiles more efficiently than the ternary operator.

32. How is comma operator useful in a for loop? **[WBSCTE 2011]**

Answer:

In the C and C++ programming languages, the comma operator is a binary operator that evaluates its first operand and discards the result, and then evaluates the second operand and returns this value (and type). The comma operator has the lowest precedence of any C operator, and acts as a sequence point. Because the comma operator discards its first operand, it is generally only useful where the first operand has desirable side effects, such as in the initialiser or the counting expression of a for loop.

33. Explain the usefulness of default statement in switch case statement. **[WBSCTE 2011]**

Answer:

The default statement can occur in any order in the switch statement. The default clause is an optional clause that is matched if none of the constants in the case statements can be matched.

34. Why we use do while loop in c? Also tell any properties which you know? **[WBSCTE 2011]**

Answer:

Do while loop is used when the body of the loop needs to be executed atleast once.

Properties :

1) do

```
{
    .
    .
    }while (<test condition>);
```

Semicolon needs to be insulted at the end of the test condition.

2) 'do' and 'while' keywords are to be used.

35. What will be the output of following code **[WBSCTE 2017]**

a)
```
main()
{
    int i=1;
    while(8)
    {
        printf("%d\n",i++);
        if(i>10)
            break;
    }
}
```

Answer: Output:

1
2
3
4
5
6
7
8
9
10

b)
```
main()
{
    int k,num=30;
    k=(num>5?(num<=10?100:200):500);
    printf("%d",k);
}
```

Answer: Output: 200

Long Answer Type Questions

1. "Use of goto statement in C language program should be avoided" – justify the statement. **[WBSCTE 2006, 2011]**

OR,

Why 'goto' statement should be avoided in a C program? **[WBSCTE 2015, 2016]**

OR,

What is goto statement and why this should be avoided? **[WBSCTE 2017]**

Answer:

All of the popular general purpose programming languages contain 'goto' statement, though modern programming practice discourages its use. The statement was used for the following:

(i) Branching around statements or groups of statements under certain conditions.

(ii) Jumping to the end of a loop under certain conditions, thus by passing the remainder of the loop during the current pass.

(iii) Jumping completely out of a loop under certain conditions, thus terminating the execution of a loop.

The structured features in C enable all of these operations to be carried out without resorting to the 'goto' statement. For example, branching around statement; jumping to the end of loop can be carried out with the continue statement; and jumping out of a loop is easily accomplished using the break statement. The use of structured features in preferable to the use of the 'goto' statement, because the use of 'goto' trends to encourage logic that skips all over the program where as the structured features in C require that the entire program be written in an orderly, sequential manner.

For this reason "use of goto statement should generally be avoided".

2. Write the differences between break and continue. **[WBSCTE 2007]**

OR,

Differentiate between 'Break' and 'Continue' statement in C with Example. **[WBSCTE 2012, 2013]**

OR,

Distinguish between 'break' and 'continue' statements. **[WBSCTE 2015]**

Answer:

Break	Continue
Break statement is used to transfer control of the program outside loop or switch case statement either conditionally or unconditionally.	Continue statement is used to skip some statement of the loop and moves to the next iteration in the loop.
Break statement is used in loop as well as Switch Case Statement.	Continue statement is used only within loop.

Example:
`for(i=1;i<=10;i++)`
`{`
`if(i%5==0)`
`break;`
`else`
`printf("%d",&i);`
`}`
Output:
1234
	Example:
	`for(i=1;i<=10;i++)`
	`{`
	`if(i%5==0)`
	`break;`
	`else`
	`printf("%d",&i);`
	`}`
	Output:
	12346789

3. Differentiate between looping and branching. **[WBSCTE 2008]**

Answer:

Branching: It is required to compute what actions to execute based on some condition.

```
if(condition)
  {
      statement;
  }
  else if(condition)
  {
      statement;
  }
  else
  {
      statement;
  }
```

Looping: It is required to compute how many times a certain action take to execute.

```
While(condition)
  {
      Statement;
  }
  OR,
  for(initialization; condition; increment)
  {
      statement;
  }
```

4. What are the advantages and disadvantages of switch statement?

Answer: **[WBSCTE 2008]**

Advantages:

(i) It leads to more structured program.
(ii) The level of indentation is manageable.
(iii) Switch case reduces complexity.

Disadvantages:

(i) The switch statement only works in integral arguments and expressions(char, the various sizes of int).
(ii) There is no direct way to switch on strings, or on floating-point values.
(iii) The target case labels must be specified explicitly; there is no general way to specify a case, which corresponds to a range of values.

5. What will be the output after execution of the following code?

[WBSCTE 2008]

```
int i=0;
while(i<5)
{
if(i<3)
{
i+=2;
```

```
printf("%d\n",i);
continue;
}
else
{
printf("%d\n",++i);
break;
}
printf("loop terminated..\n");
}
```

Answer:

The output of the code is →

2
4
5

Initially i=0; while loop condition i<5 is satisfied.
Also the if condition i<3 is satisfied.
Now, i becomes i=2, which is printed in the next line.
'continue' statement redirects the control of the program to the beginning of the while loop.
i=2, which is less than 3, therefore if condition is satisfied, and the value of i is increased by 2 and becomes i=4. 4 is printed in the next line and again the control goes to if condition checking line due to continue statement. But, now i=4, so if condition is not satisfied, and else block is executed. In the else block, the value of i is increased by 1, and printed. The break statement in the next line causes exit from while loop; so "loop terminated" is not printed.

6. What is a control statement? Give example. **[WBSCTE 2009]**

Answer:

All statements written in a program are executed from top to bottom one by one. Control statements are used to execute/transfer the control from one part of the program to another, depending on a condition. These statements are also called as conditional statements. There are two types of control conditional statements used in C language – (a) if-else statement, (b) switch statement.

7. Use of goto statement in C language program should be avoided" – justify the statement. **[WBSCTE 2009]**

Answer:

All of the popular general purpose programming languages contain 'goto' statement, though modern programming practice discourages its use. The statement was used for the following:

(i) Branching around statements or groups of statements under certain conditions.
(ii) Jumping to the end of a loop under certain conditions, thus by passing the remainder of the loop during the current pass.
(iii) Jumping completely out of a loop under certain conditions, thus terminating the execution of a loop.

The structured features in C enable all of these operations to be carried out without resorting to the 'goto' statement. For example, branching around statement; jumping to the end of loop can be carried out with the continue statement; and jumping out of a loop is easily accomplished using the break statement. The use of structured features in preferable to the use of the 'goto' statement, because the use of 'goto' trends to encourage logic that skips all over the program where as the structured features in C require that the entire program be written in an orderly, sequential manner.

For this reason "use of goto statement should generally be avoided".

8. (a) Discuss the usefulness of Jumping in loops. How are they implemented in C?

(b) What are the various control structures used in structured programming? [WBSCTE 2010]

Answer:

a) The jumping statement in a loop is a unidirectional transfer of control statement. It is used to transfer the control from one part of the program to another –

statements;
jump display;
statements;
statements;
display;
statements;

Jumping is executed in C using goto statement.

b) The following control structures are used in structured programming –

(i) if-else block,
(ii) switch-case block,
(iii) for loop,
(iv) while loop,
(v) do-while loop.

(i) if-else block structure:

```
if(expression)
        statement;
else
        statement;
```

(ii) switch-case block structure:

```
switch(expression)
{
        case c1:        statement;
        case c2:        statement;
        ..................
        ..................
        default:        statement;
}
```

(iii) for loop structure:

```
for(initialization; condition; increment)
{
        statement;
}
```

(iv) while loop structure:

```
while(condition)
{
        statement;
}
```

(v) do-while loop structure;

```
do
{
        statement;
}while(condition);
```

9. (a) Give an example to show the functionality of switch-case.
(b) What is loop? What are the different loop construct in C?
(c) Write a code to show infinite loop works effectively in C? **[WBSCTE 2010]**

Answer:

a)

```
/**** program for a simple calculator****/
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        float x, y, ans=0;
        charsym;
        printf("\nEnter the value of x and y: ");
        scanf("%f %f",&x,&y);          /*Input for x and y*/
        printf("\nEnter an operator:");
        scanf("%c",&sym);      /* Input for operator*/

        /****Start of switch case****/
        switch(sym)            /*Switch case condition check*/
        {
                case '+':      ans = x+y;     /*If input operator is '+', this
case block is executed*/
                                    break;
                case '-':      ans = x-y;     /*If input operator is '-', this
case block is executed*/
                                    break;
                case '*':      ans = x*y;     /*If input operator is '*', this
case block is executed*/
                                    break;
```

```
            case '/':       ans = x/y;   /*If input operator is '/', this
case block is executed*/
                                    break;
default:        ans =0;         /*If input oerator does not match '+' or '-' or
'*' or '/', this block is executed*/
                                    printf("This is not write operator.");
                                    break;

        }
        /****End of switch case****/

        printf("\nAnswer = %f", ans);          /*Print the answer.*/
        getch();
}
```

b) When a program performs same series of action in the same way, then it is called loop. There are three type of loop generally used in C programming. These are –
(i) while loop, (ii) do-while loop, (iii) for loop.

Different loop construction in C:

while loop →

```
while(condition)
{
        statements;
}
```

do-while loop →

```
do
{
        statements;
}while(condition);
```

for loop →

```
for(initialization; condition; increment/decrement)
{
        statements;
}
```

c) Sometimes infinite loop works efficiently in C program. To evaluate the value cosine or sine, to find proper root of a fractional number, to sense whether a key is pressed or not, we need a loop, which can run infinitely and work efficiently in C programming. Here, the infinite loop checks, when a key is pressed from keyboard:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr();
while(1)        /*loop condition is always true*/
{
if(kbhit( ) )   /*Checks whether a key is pressed or not*/
printf("A key is pressed.\n");
}
}
```

10. With example explain the functionality of continue. **[WBSCTE 2010]**

Answer:

Sometimes during iteration of a loop a particular statement execution is deferred until a suitable condition is reached. This situation can be handled by a combination of an if statement and a continue statement.

Example →

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int i;
for (i=1; i<=5; i++)
{
if(i%2==0)
continue;
printf("%2d", i);
}
getch( );
}
```

The output of the program is → 1 3 5

Whenever the condition i%2==0 is true, the printf is not executed rather the continue statement returns back the control to the for loop.

11. Write a C program to print the sum of all odd numbers from 1 to 100. **[WBSCTE 2011]**

Answer:

```
#include <stdio.h>
#include <conio.h>
void main()
{
      int i,sum=0;
      for (i=1;i<100;i+=2)
            sum+=i;
printf("sum is %d", sum);
getch();
}
```

12. Write a C program to evaluate a given number is palindrome or not **[WBSCTE2011]**

Example: **i/p-> 12321 it is palindrome number.**
i/p-> 143 it is not

Answer:

```
#include <stdio.h>
#include <conio.h>
#include<stdlib.h>
int main(void)
{
```

```
  long int x, y, n;
  int i;
  printf("Enter a number\n");
  scanf("%ld", &x);
  y = 0;
  n = x;
  while (x != 0)
  {
    i = x % 10;
    x = x / 10;
    y = y * 10+i;
  }
  if (n == y)
  {
    printf("The number% ld is a palindrome", n);
  }
  else
  {
    printf("The number% ld is not a palindrome", n);
  }
  getch();
  return EXIT_SUCCESS;
}
```

13. Predict the output or error(s) for the following and explain **[WBSCTE 2011]**

```
main( )
   {
float me = 1.1;
double you = 1.1;
if(me==you)
printf("I love U");
else
printf("I hate U");
   }
```

Answer:

Output	Explanation
I hate U	As the variables '*me*' and '*you*' are of different datatypes, the test condition turns out to be false and the else part is executed.

14. Write a C program to print the following Pattern using loop. **[WBSCTE 2011]**

```
      *
    *   *
  *       *
* * * *
```

Answer:

```
#include<stdio.h>
#include<conio.h>
```

```
void main()
{
clrscr();
int i,j;
for (i=1;i<=3;i++)
{
for(j=4-i;j>0;j--)
printf(" ");
printf("*");
for(j=i-1;j>0;j--)
printf(" ");
for(j=i-2;j>0;j--)
printf(" ");
if(i!=1)
printf("*");
printf("\n");
}
for(i=1;i<=4;i++)
printf("* ");
getch();
}
```

15. Explain the importance of a switch case statement. In which situation is a switch case desirable? Also give its limitations. **[WBSCTE 2011]**

Answer:

Switch case statements are a substitute for long if statements that compare a variable to several "integral" values ("integral" values are simply values that can be expressed as an integer, such as the value of a char). The basic format for using switch case is outlined below. The value of the variable given into switch is compared to the value following each of the cases, and when one value matches the value of the variable, the computer continues executing the program from that point.

```
switch ( <variable> ) {
case this-value:
  Code to execute if <variable> == this-value
  break;
case that-value:
  Code to execute if <variable> == that-value
  break;
...
default:
  Code to execute if <variable> does not equal the value
following any of the cases
  break;
}
```

Limitation:

Switch statement is a powerful statement used to handle many alternatives and provides good presentation for C program. But there are some limitations with switch statement which are given below:

Logical operators cannot be used with switch statement. For instance

case k>=20:

is not allowed

Switch case variables can have only int and char data type. So float or no data type is allowed.

For instance in the switch syntax given below:

```
switch(ch)
{
case 1:
statement-1;
break;
case 2:
statement-2;
break;
}
```

In this ch can be integer or char and cannot be float or any other data type.

16. Predict are output or error(s) for the following and explain. **[WBSCTE 2011]**

```
main()
  {
        static int var = 5;
        printf("%d",var--);
        if(var)
        main();
  }
```

Answer:

Output: 5 4 3 2 1

Explanation:

When static storage class is given, it is initialized once. The change in the value of a static variable is retained even between the function calls. Main is also treated like any other ordinary function, which can be called recursively.

17. Why we use do-while loop in C? **[WBSCTE 2011]**

Answer:

It is also called as post tested loop. It is used when it is necessary to execute the loop at least one time.

Syntax:

do {

Loop body

} while (Expression);

18. Could you define variable in c? **[WBSCTE 2012, 2013]**

Answer: int k;

k=3;

the first statement is declaration and the second statement is assignment .Both constitutes definition of variable k in c.

19. What is difference between initialization and assignment of a variable? [WBSCTE 2012, 2013]

Answer:
int i=21; this contains a value 21 at the time of creation of variable i of integer type in c, whereas int k; k contains garbage value at the time of creation of variable i. k=21. k is assigned a value 21 later.

20. What is the difference between a statement and a block? [WBSCTE 2012, 2013]

Answer:
a statement ends with a semicolon and the compiler sees it a single unit at the time of execution. Whereas a group of statements within second braces form a block;

21. Predict the output or error(s) for the following: Explain. [WBSCTE 2012]

```
void main() {
    int i;
    for(i=0;i<=5;i++);
            printf("%d",i);
    getch();
}
```

Answer:
i =0 |1|2|3|4|5| - |
i++ =1 |2|3|4|5|6| - |
it will print 6.
Answer is 6

22. (a) Write a C-program to print whether a number is palindrome or not.
(b) Write a C-program to convert any decimal number to its equivalent binary number. [WBSCTE 2013]

Answer:
a) /* C program to check whether a number is palindrome or not */

```
#include <stdio.h>
int main()
{
int n, reverse=0, rem,temp;
printf("Enter an integer: ");
scanf("%d", &n);
temp=n;
while(temp!=0)
{
rem=temp%10;
reverse=reverse*10+rem;
temp/=10;
}
/* Checking if number entered by user and it's reverse number is
equal. */
if(reverse==n)
```

```
printf("%d is a palindrome.",n);
else
printf("%d is not a palindrome.",n);
return 0;
}
```

b)
```
#include<stdio.h>
int main(){
long int decimalNumber,remainder,quotient;
int binaryNumber[100],i=1,j;
printf("Enter any decimal number: ");
scanf("%ld",&decimalNumber);
quotient = decimalNumber;
while(quotient!=0){
binaryNumber[i++]= quotient % 2;
quotient = quotient / 2;
}
printf("Equivalent binary value of decimal number %d:
",decimalNumber);
for(j = i -1 ;j> 0;j--)
printf("%d",binaryNumber[j]);
return 0;
}
```

➲ 23. Write a C-program to find the GCD (Greatest Common Divisor) of two numbers. **[WBSCTE 2013]**

Answer:

```
#include<stdio.h>
int main(){
int x,y,m,i;
printf("Insert any two number: ");
scanf("%d%d",&x,&y);
if(x>y)
m=y;
else
m=x;
for(i=m;i>=1;i--){
if(x%i==0&&y%i==0){
printf("\nHCF of two number is : %d",i) ;
break;
}
}
return 0;
}
```

➲ 24. Do you know nested loop. Explain by an example. **[WBSCTE 2013]**

Answer:

Nested loops are executed in order. The inner loop is executed then the outer loop is executed. However it increases the time complexity of the program.

```
for(i=0;i<5;i++){
```

```
for(j=0;j<5;j++){
printf("i=%d j=%d", i,j);// for i=0 the inner j is executed 5
times..so on
}
}
```

25. What is the difference between a statement and a block? **[WBSCTE 2013]**

Answer:

A statement ends with a semicolon and the compiler sees it a single unit at the time of execution. Whereas a group of statements within second braces form a block;

26. How is a comment formed in C language? **[WBSCTE 2013]**

Answer:

A comment is ignored by the compiler. We use //comment or /* comment*/ in c.

27. What is type of checking? **[WBSCTE 2013]**

Answer:

At the time of compilation the type of variable holds the value is checked. The lvalue is checked with r value to see overflow or any kind of type mismatch.

28. Differentiate between if else and Switch case statement in C with Example. **[WBSCTE 2013, 2016]**

Answer:

If we use an "if statement" in an "if statement" it is called nesting. Nesting "if statements" can make a program very complex, but sometimes there is no other way. We take a look at a nested "if statement" example below:

```
#include<stdio.h>
int main()
{
int grade;
scanf("%d",&grade);
if ( grade <= 10 )
{
printf("YOU DID NOT STUDY.\n");
printf("YOU FAILED ! \n");
}
else
{
if ( grade < 60 )
{
printf("You failed \n");
      printf("Study harder\n");
}
else
{
if ( grade >= 60 )
printf("YOU PASSED ! \n");
}
```

```
}
return 0;
}
```

So let's walk through the example above: If the grade is equal or below ten, you did not study. If the grade is above ten, the program will go into the "else statement". In the "else statement" there is another "if statement" (this is nesting).

This "if statement" checks the grade again. If the grade is below sixty, we must study harder. In the "else statement" from this "if statement" there is another "if statement". It checks whether the grade is above or equal to sixty. If it is, you passed the test.

The switch statement is almost the same as an "if statement". The switch statement can have many conditions. We start the switch statement with a condition. If one of the variable equals the condition, the instructions are executed. It is also possible to add a default. If none of the variable equals the condition the default will be executed.

Let's see the example below:

```
#include<stdio.h>
int main()
{
char myinput;
printf("Which option will you choose:\n");
printf("a) Program 1 \n");
printf("b) Program 2 \n");
scanf("%c", &myinput);
switch (myinput)
{
case 'a':
printf("Run program 1\n");
break;
case 'b':
{
printf("Run program 2\n");
printf("Please Wait\n");
break;
}
default:
printf("Invalid choice\n");
break;
    }
    return 0;
    }
```

29. (a) Write a program in C to find the sum of the series $1^2+(1^2+2^2)+(1^2+2^2+3^2)+\ldots$ up to n terms. **[WBSCTE 2015]**

(b) Distinguish between entry controlled and exit controlled loop structure mentioning at least three points. **[WBSCTE 2015, 2016]**

Answer:

a)

```
#include <stdio.h>
```

```
ıclude <conio.h>
.in()
 int i,j,s1,s2=0;
 Clrscr();
printf("Enter the limit");
scanf("%d",&n);
for(i=1; i<= n; i++)
{ s1=0;
for (j=1;j<= i; j++)
 s1= s1+ i*i ;
 s2 = s2 + s1;
}
printf("The final summation value is %d" , s2);
getch();
}
```

b)

No.	Topics	Entry controlled loops	Exit controlled loops
01	Test condition	Test condition appears at the beginning	Control variable is counter & sentinel variable.
02	Control variable	Control variable is counter variable	Each execution except the first one occurs by testing condition.
03	Execution	Each execution occurs by testing condition	

04	Examples	========
		sum = 0;
		n = 1;
		while (n <= 10)
		{
		sum = sum + n*n;
		n = n + 1;
		}
		=========
			======
			do
			{
			printf("Input a number.\n");
			scanf(*%d*,&num);
			}
			while(num>0);
			======

➲ 30. Write a program in C to find all prime numbers within a given range. **[WBSCTE 2015, 2016]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
   int r1,r2,i,n,flag,s;
   clrscr();
   printf("Enter Start and End Range : ");
   scanf("%d%d",&r1,&r2);
   printf("All Prime number from %d to %d \n",r1,r2);
```

```
    for(n=r1;n<=r2;n++)
    {
    flag =0;
    s=sqrt(n);
    for(i=2;i<=s;i++)
    if(n%i==0)
    {
          flag =1;
          break;
    }
    if(flag==0)
          printf("%5d",n);
    }
    getch();
}
```

31. Write a program in C to find the sum of the series 1+(1+2)+(1+2+3)+..... up to 'n' terms, the value of 'n' should be given from the keyboard.

[WBSCTE 2016]

Answer:

```
#include <stdio.h>
#include <conio.h>
main()
{ int i,j,s1,s2=0;
  clrscr();
printf("Enter the limit");
scanf("%d",&n);
for(i=1; i<= n; i++)
{ s1=0;
for (j=1;j<= i; j++)
 s1= s1+ i ;
 s2 = s2 + s1;
}
printf("The final summation value is %d" , s2);
getch();
}
```

32. What is the difference between while loop and do-while loop? Explain with example.

Answer: **[WBSCTE 2017]**

1) While loop

A **while** loop statement repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a while loop in C is –

```
while(condition) {
   statement(s);
```

ere, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop. It is termed as **Entry controlled** loop.

Example:

```
#include <stdio.h>
#include <conio.h>
int main () {
// Local variable declaration:
int a = 10;
// while loop execution
while( a < 20 ) {
printf("%d ",a);
a++;
}
return 0;
}
```

2) do while loop

Unlike **for** and **while** loops, which test the loop condition at the top of the loop, the **do...while** loop checks its condition at the bottom of the loop.

Syntax

The syntax of a do...while loop in C is –

```
do {
   statement(s);
}
while( condition );
```

A **do...while** loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time. Therefore it is termed as **Exit controlled** loop. If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop execute again. This process repeats until the given condition becomes false.

Example:

```
#include <iostream>
using namespace std;
int main () {
// Local variable declaration:
int a = 10;
// do loop execution
do {
printf("%d",a);
a = a + 1;
} while( a < 20 );
return 0;
}
```

33. a) Differentiate between if-else if and switch statement.
b) Write a program to check if a character entered through keyboard is a vowel or consonant using switch case statement.

[WBSCTE 2017]

Answer:

a) In **if – else-if** control we can provide two conditions as per user's requirement, firstly if condition will be checked if it is true then the if block will be executed, if false then else-if condition will be checked and if it is true then the else-if block will be executed.

The **switch** statement can have many conditions. We start the switch statement with a condition. If one of the variables equals the condition, the instructions are executed. It is also possible to add a default. If none of the variable equals the condition the default will be executed.

b)

```
#include <stdio.h>
#include <conio.h>
main( )
{
      char ch;
      clrscr( );
      printf("\n Enter a character to check: ");
      scanf("%c", &ch);
      switch(ch)
      {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U':
      printf("%c is a vowel", ch);
      break;
            default:
      printf("%c is not a vowel", ch);
      break;
getch( );
      }
}//End of main( )
```

Unit 3: Arrays and Strings

Unit at a glance:

3.1 Introduction

An array is a finite set of ordered, homogeneous elements. It is a linear data structure, in which consecutive memory locations are used to store the data. **Single-Dimension Array:**

In C, one-dimensional arrays of different data types are declared in the following manner.

```
int arr [ 6 ] = { 4 , 5 , 2 0 , 99 ,14 ,6 } ; / * integer array */
float arr [ 5 ] = { 4 . 5 , 6 . 7 , 8 . 3 , 9 . 1 , 1 0 . 2 } ; / * float array */
char arr [ 5 ] = { 'A' , 'B' 'C' , 'D' , 'E ' } ; / *character array */.
```

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. An array cannot be initialized provided it is – **[WBSCTE 2007, 2009, 2010]**
   (a) automatic (b) external (c) static (d) both b & c

Answer: (d)
(Automatic variable can be initialized later.)

2. The string terminator in C-language is – **[WBSCTE 2007, 2009,2017]**
   (a) '\b' (b) '\&' (c) '\c' (d) '\0'

Answer: (d)

3. The library function strcmp("abc","ABC"); returns – **[WBSCTE 2007, 2009]**
   (a) 32 (b) -32 (c) 1 (d) -1

Answer: (a)
(Since the difference between 'a's ASCII value and 'A's ASCII value is 32)

4. Which is more appropriate for reading in a multiword string? **[WBSCTE 2010]**
   (a) gets( ) (b) fgets( ) (c) scanf( ) (d) scan( )

Answer: (a)

5. char name [ ] = {'T', 'W', 'D','T', 'A', '\0'};
   printf("%d", strlen (name));
   The output is – **[WBSCTE 2010]**
   (a) 6 (b) 0 (c) 5 (d) 7

Answer: (a)

6. What is the output? [WBSCTE 2019]

```
char a[ ]= " ABCD", b[ ]= "abcd";
strcpy(b,a);
printf("%s %s", a, b);
```

(a) abcd ABCD (b) ABCD ABCD (c) ABCD abcd (d) abcdabcd

Answer: (b)

(strcpy(str1, str2) copies 1st string to the 2nd)

7. In C, if you pass an array as an argument to a function, what actually gets passed? [WBSCTE 2011]

(a) Value of elements in array (b) First of elements of the array
(c) Base address of the array (d) Address of the last element of array

Answer: (c)

8. What will happen if in a C program you assign a value to an array element whose subscript exceeds the size of array? [WBSCTE 2011]

(a) The element will be set to 0
(b) The compiler would report an error
(c) The program may crash if some important data gets overwritten
(d) The array size would appropriately grow

Answer: (c)

9. If the two strings are identical, then *strcmp*( ) function returns- [WBSCTE 2011]

(a) –1 (b) 1 (c) 0 (d) Yes

Answer: (c)

10. Array passed as argument to a function is treated as: [WBSCTE 2013]

(a) Address of the array
(b) Value of the first element of the array
(c) Address of the first element of the array
(d) Number of elements of the array

Answer: (c)

11. The library function strcmp("abc","ABC"); returns [WBSCTE 2015]

(a) 32 (b) –32 (c) 0 (d) 1

Answer: (d)

12. Array is collection of [WBSCTE 2017]

(a) different data types scattered throughout memory
(b) the same data type scattered throughout memory
(c) the same data type placed next to each other in memory
(d) different data types placed next to each other in memory

Answer: (c)

13. If an array is declared as arr[]={1,2,5,7,9}; then what is the value of size of (arr[3])? [WBSCTE 2017]

(a) 1 (b) 2 (c) 3 (d) 8

Answer: None of these. (7 is the answer)

14. s1="H1", s2="HELLO", s3="BYE". How can we concatenate the three strings? [WBSCTE 2017]

(a) strcat(s1,s2,s3)
(b) strcat(s1,strcat(s2,s3))
(c) strcpy(s1,s2,s3)
(d) strcpy(s1,strcpy(s2,s3))

Answer: (b)

B. State whether the following statements are True or False:

15. In C language pointer is basic or fundamental data type. [WBSCTE 2007]

Answer: False
(Basic or fundamental data types are – int, float, char, double etc.)

16. The declaration int x[2]={1,2,3}; is illegal. [WBSCTE 2008]

Answer: True

17. In C, by default the first subscript is zero. [WBSCTE 2008]

Answer: True

18. When initializing a multidimensional array, not specifying all its dimensions is an error. [WBSCTE 2009]

Answer: True (No. of columns must be specified).

19. In C, strings cannot be initialized at run time. [WBSCTE 2009]

Answer: False (Strings are to initialized at compile time)

20. The function call strcmp("abc", "ABC"); returns a positive number. – [WBSCTE 2009]

Answer: True (Returns 32)

21. The code 'a[i]=i++' will produce an error. [WBSCTE 2010]

Answer: False

22. C uses 'column major representation' for storing multidimensional array in memory. [WBSCTE 2010]

Answer: True

23. The code 'int a[5] ={0}' will initialize a[4] to be a garbage value. [WBSCTE 2010]

Answer: False (a[4] will be initialized to zero.)

24. The function call strcmp("abc","ABC"); will return a positive number. [WBSCTE 2010]

Answer: True (It will return 32)

25. The function call strcmp ("abc", "ABC"); will return a zero. [WBSCTE 2013]

Answer: False

C. Answer the following questions:

26. **Write the statement to increment the i-th element of an array named 'ar'.** [WBSCTE 2008]

Answer:
The statement to increment the i-th element of an array named 'ar' –
ar [i]++;

27. What is the objective of the function strcpy (S1, S2)? [WBSCTE 2008, 2009]
Answer: strcpy (S1, S2) copies string S2 to string S1.

28. Write the statement to copy the contents of the i-th element of an array to the k-th element of that array. [WBSCTE 2008, 2010]

Answer:
The statement to copy the contents of the i-th element of an array to the k-th element of that array –

ar [k] = ar [i];

29. What is the difference between arr and &arr, where arr is an array name? [WBSCTE 2008]

Answer:
arr and &arr both store the base address of array named arr.

30. What will be the output of the following program? [WBSCTE 2009]

```
main( )
{
    int sub[50], i;
    for (i=0;i<=48;i++);
    {
    Sub[i]=I;
    printf("\n%d", sub[i]);
    }
}
```

Answer:
The output will be 49. Because, there is a semicolon at the end of for statement. So, value of i will start from 0 and will increase to 49 without executing the body of the loop. After that the loop will terminate at i = 49. The next statement will be executed and sub[49] = 49. So, 49 will be printed.

31. How is an array name interpreted? [WBSCTE 2010]
Answer:
An array name is interpreted as the address of the starting location of the array.

32. Explain the error in the following code –

int a[ ]={10,20,30,40,50};
a++;

[WBSCTE 2010]
Answer: As it is an array, so, by 'a++' statement the increment of all terms is not possible. a[ ] is a sets of integer value. So it is an error.

33. Explain when strcmp( ) returns negative value. [WBSCTE 2010]
Answer: strcmp( ) function is used to compare two character strings. It returns a 0 when two strings are identical; otherwise it returns a numerical value. When the first string is alphabetically before the second string, a negative value is returned.

34. A 2 dimensional array is declared as float a[10][20]. What is the address of a[8][14] provided the base address of a is 20000. **[WBSCTE 2010]**

Answer:

A 2 dimensional array is declared as float a[10][20].

Base address is 20000.

Therefore, a[8][14] address is $\{(8 \times 21 + 15) \times 4 + 20000\} = 20732$.

35. How is an array represented in memory? **[WBSCTE 2011, 2017]**

Answer:

As we know now 1-d array are linear array in which elements are stored in the successive memory locations. The element at the very first position in the array is called its base address. Now consider the following example :

int arr[5];

34	78	98	45	56
arr[0] = 100	arr[1]	arr[2]	arr[3]	arr[4]

Here we have defined an array of five elements of integer type whose first index element is at base address 100. That is, the element arr[0] is stored at base address 100. Now for calculating the starting address of the next element i.e. of a[1], we can use the following formula :

Base Address (B)+ No. of bytes occupied by element (C) * index of the element (i)

/* Here C is constant integer and vary according to the data type of the array, for e. integer the value of C will be 4 bytes, since an integer occupies 4 bytes of memory. *

Now, we can calculate the starting address of second element of the array as :

arr[1] = 100 + 4 * 1 = 104 /*Thus starting address of second element of array is 104

Similarly other addresses can be calculated in the same manner as :

arr[2] = 100 + 4 * 2 = 108

arr[3] = 100 + 4 * 3 = 112

arr[4] = 100 + 4 * 4 = 116

36. What do you mean by type casting in C? **[WBSCTE 2012]**

Answer:

Type Casting is a method using which a variable of one data type is converted to another data type.

37. Why do we need signed and unsigned integer? **[WBSCTE 2013]**

Answer:

signed integer holds signed data. The arithmetic instructions like substraction and overflow detection is done with signed integer. Unsigned integer is used to hold positive integers only.

38. Why should we not use floating point numbers for equality in expressions? **[WBSCTE 2013]**

Answer:

Rounding error is inherent in floating-point computation.Consider the floating-point format with $\beta = 10$ and $p = 3$. If the result of a floating-point computation is 3.12×10^{-2},

and the answer when computed to infinite precision is .0314, it is clear that this is in error by 2 units in the last place. Similarly, if the real number .0314159 is represented as 3.14×10^{-2}, then it is in error by .159 units in the last place.

39. What does array name specify? [WBSCTE 2013]

Answer:
Array name is the name given to the memory location that holds the contiguous block of same type of data.

40. Write the output of the following program fragment: [WBSCTE 2013]

```
main()
{
        printf("Examination");
}
```

Answer: Examination

41. Write the output of the following program fragment: [WBSCTE 2013]

```
main()
{
        int A=3;
        printf("A");
}
```

Answer: A

42. Distinguish between gets(name) and scanf("%s", name) where 'name' is a character array. [WBSCTE 2016, 2017]

Answer:
The main difference between them is that gets allows to read white spaces but scanf does not allow. So gets just reads the line and scanf parses the line and then reads the data into one or more variable according to the format string.

Long Answer Type Questions

1. Write a C program which will delete all space in between two words of a string or sentence taken as input from the keyboard. [WBSCTE 2007]

Answer:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int i, j=0;
char s[25], a[25];
printf("Enter a string:");
gets(s);
for(i=0; s[i]!='\0'; i++)
{
```

```
if(s[i]!=' ')          //Check for whitespace
{
a[j]=s[i];
j++;
}
}
printf("The string without spaces: %s",a);
getch( );
}
```

2. Differentiate between calloc ( ) and malloc () functions in C.

[WBSCTE 2008, 2011]

Answer:

malloc( ): The function most commonly used for dynamic memory allocation is malloc( ). The prototype of the function is given below:

void * malloc (size_t size)

malloc returns a pointer to space for an object of size size, or NULL if the request cannot be satisfied. The syntax of the function is given below:

malloc (b),

where b is an unsigned integer which stands for the number of bytes you want to draw from heap. The function returns a pointer, if your request is successful, NULL otherwise.

calloc( ): Function calloc( ), like malloc( ) allocates memory in a dynamic manner. The prototype of the function is given below:

void * calloc (size_t nobj, size_t size)

calloc returns a pointer to space for an array of nobj objects, each of size size, or NULL if the request cannot be satisfied. It takes, as arguments, two values as given below:

p = (t *) calloc(n, b);

(t*) does the type casting to type t as in case of malloc( ). The above statement, allocates n blocks of memory, each block with b bytes. This function is useful for storing arrays, etc. If each element of an array requires b bytes and the array is required to store k elements, we can allocate memory by the following statement:

p = (t*) calloc (k, b);

The same allocation will done by malloc( ), if we call the function as:

p = (t*) malloc(k*b);

3. Write a C program to find the largest and smallest number of a matrix.

[WBSCTE 2009]

Answer:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int a[4][4], max, min, i, j;
printf("Enter a 4x4 arix:");
/*Enter for matrix*/
for(i=0; i<4; i++)
```

```
{
for(j=0; j<4; j++)
scanf("%d", &a[i][j]);
}
printf("The matrix you have entered:\n");
for(i=0; i<4; i++)
{
for(j=0; j<4; j++)
{
printf("%d\t", a[i][j]);
}
printf("\n");
}
max=a[0][0];
min=a[0][0];
/*Find for maximum and minimum number*/
for(i=0; i<4; i++)
{
for(j=0; j<4; j++)
{
if(a[i][j]>max)
max=a[i][j];
else if(a[i][j]<min)
min=a[i][j];
}
}
printf("\nMaximum:%d\nMinimum:%d",max, min);
getch( );
}
```

➲ 4. "C does not do boundary checking on the elements of an array." — Do you agree with this statement? Explain how an array reference is resolved by C. **[WBSCTE 2009]**

Answer:

Yes, C programming language does not perform array bounds checking.
It means it doesn't check to see if the array element you're trying to access is even part of the array.
Assume that you have an array with 10 elements called "foo".
foo[20] is legal in C, because it doesn't check the bounds.

➲ 5. Write a program to reverse a string using a function call. **[WBSCTE 2009]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void reverse (int index, char *str ) ;
int main (void)
{
char name[100];
```

```
printf ("Enter a mutli-word string") ;
gets(name) ;
reverse (strlen(name) , name ) ;
}
void reverse (int index, char *str )
{
if (--index < 0 )
{
return ;
}
else
{
putchar ( *(str + index) ) ;
reverse (index, str) ;
}
}
```

6. Write a C program to arrange a set of n numbers in ascending order. **[WBSCTE 2010]**

Answer:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
inti,j,a[10], n;
clrscr();
printf("Enter the no. of elements(Max. 10):");
scanf("%d", &n);
printf("Enter elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for(j=0;j<(2*n);j++)
{
for(i=0;i<n-1;i++)
{
if(a[i]>a[i+1])
{
a[i]=a[i+1]+a[i];
a[i+1]=a[i]-a[i+1];
a[i]=a[i]-a[i+1];
}
}
}
printf("\nSorted elements(in ascending order:");
for(i=0;i<n;i++)
printf("%d\t",a[i]);
getch( );
}
```

7. Write a C program that will delete all spaces between the words of a sentence. **[WBSCTE 2010]**

Answer:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int i, j=0;
char s[25], a[25];
printf("Enter a string:");
gets(s);
for(i=0; s[i]!='\0'; i++)
{
if(s[i]!=' ')            //Check for whitespace
{
a[j]=s[i];
j++;
}
}
printf("The string without spaces: %s",a);
getch( );
}
```

8. How do you read the elements of a n × n matrix? Also print the elements replacing non-diagonal elements with the characters '*'. **[WBSCTE 2010]**

Answer:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
int a[10][10], i, j, n;
printf("Enter the no. of elements(n)[Max. 10]:");
scanf("%d", &n);
printf("Enter the nxn matrix:");
/*Enter for matrix*/
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
scanf("%d", &a[i][j]);
}
printf("The matrix you have entered:\n");
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
{
printf("%d\t", a[i][j]);
}
printf("\n");
}
printf("\nThe matrix with only diagonal elements:\n");
```

```
for(i=0; i<n; i++)
{
for(j=0; j<n; j++)
{
if(i==j || j==(n-1)-i)          ////For diagonal elements
printf("%d\t", a[i][j]);
else                            ////For non-diagonal elements
printf("*\t");
}
printf("\n");
}
getch( );
}
```

9. (a) Write a C program to simulate strlen( ).
(b) Write a C program to append a new string at the end of an existing string.
[WBSCTE 2010]

Answer:

a)
```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
charstr[50];
int i=0, len=0;
printf("Enter a string:");
gets(str);
while(str[i]!='\0')
{
len++;
i++;
}
printf("Length of the string:%d", len);
getch( );
}
```

b)
```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
char str1[25], str2[25], str[50];
int i=0, j=0;
printf("Enter first string:");
gets(str1);
printf("Enter second string:");
gets(str2);
for(i=0; str1[i]!='\0'; i++)
{
str[i]=str1[i];
```

```
}
for(j=0; str2[j]!='\0'; j++)
{
str[i]=str2[j];
i++;
}
str[i]='\0';
printf("\nConcatenated string is:");
puts(str);
getch( );
}
```

10. Differentiate Static and Dynamic memory allocation. [WBSCTE 2011]

Answer:

Static Memory Allocation	Dynamic Memory Allocation
Memory is allocated before the execution of the program begins. (During Compilation)	Memory is allocated during the execution of the program.
No memory allocation or deallocation actions are performed during Execution.	Memory Bindings are established and destroyed during the Execution.
Variables remain permanently allocated.	Allocated only when program unit is active.
Implemented using stacks and heaps.	Implemented using data segments.
Pointer is needed to accessing variables.	No need of Dynamically allocated pointers.
Faster execution than Dynamic.	Slower execution than static.
More memory Space required.	Less Memory space required.

11. Write a C program to the sum of diagonal elements of a square matrix. [WBSCTE 2012]

Answer:

```
#include<stdio.h>
int main(){
int a[10][10],i,j,sum=0,m,n;
printf("\nEnter the row and column of matrix: ");
scanf("%d %d",&m,&n);
printf("\nEnter the elements of matrix: ");
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
printf("\nThe matrix is\n");
for(i=0;i<m;i++){
printf("\n");
for(j=0;j<m;j++){
printf("%d\t",a[i][j]);
}
}
for(i=0;i<m;i++){
for(j=0;j<n;j++){
if(i==j)
sum=sum+a[i][j];
}
}
```

```
printf("\n\nSum of the diagonal elements of a matrix is:
%d",sum);
return 0;
}
```

12. How do you read the elements of a n x n matrix? What is an array? [WBSCTE 2013]

Answer:

To read the elements of nxn matrix we can use two for loops .

For example, Int array[3][3];

```
printf("enter the values of nxn matrix");
for(i=0;i<3;i++){
for(j=0;j<3;j++){
scanf("%d %d",&a[i][j]);
}
}
```

An array in C Programing Language can be defined as number of memory locations, each of which can store the same data type and which can be references through the same variable name.

13. (a) How do you read the elements of anxn matrix? What is an array?
(b) Write a C-program to find the GCD (Greatest Common Divisor) of two numbers. [WBSCTE 2013]

Answer:

a) To read the elements of nxn matrix we can use two for loops .

For example, Int array[3][3];

```
printf("enter the values of nxn matrix");
for(i=0;i<3;i++){
for(j=0;j<3;j++){
scanf("%d %d",&a[i][j]);
}
}
```

An array in C Programing Language can be defined as number of memory locations, each of which can store the same data type and which can be references through the same variable name.

b)

```
#include<stdio.h>
int main(){
int x,y,m,i;
printf("Insert any two number: ");
scanf("%d%d",&x,&y);
if(x>y)
m=y;
else
m=x;
for(i=m;i>=1;i--){
if(x%i==0&&y%i==0){
printf("\nHCF of two number is : %d",i) ;
break;
}
```

```
}
return 0;
}
```

14. Write C programs to implement i) strcmp(), ii) strcat() library function by yourself. **[WBSCTE 2015]**

Answer:

```
#include <stdio.h>
#include <string.h>
main()
{
char s1[50], s2[50];
strcpy( s1, "Be" );
strcpy( s2, "Happy" );
strcat( s1, s2 );
puts( s1 );
if ( strcmp( s1,s2 ) = = 0 )
puts( "same string" );
else
puts(" two different string");
getch()
}
}
```

Here strcpy(s1, "Be") and strcpy(s2, "Happy") will copy words 'Be' and 'Happy' into string s1 and s2 respectively. Then strcat(s1, s2) will concatenate them together and output will be 'BeHappy'. As these two strings are not same so strcmp(s1,s2) will give the output 'two different string'.

15. Define a structure to store the name, roll, an array marks[] which stores marks of three different subjects and a character grade on total marks. Write a program to display the details of the 'n' (given from the keyboard) student in tabular form with grade as per rules – (i) 75% and above Grade 'A', (ii) 60% to less than 75% Grade 'B', (iii) Less than 60% Grade C. **[WBSCTE 2015]**

Answer:

```
#include<stdio.h>
#include<conio.h>
void main()
{
struct student
{
int rollno;
char name[20];
int marks[4];
int total;
float percent;
};
struct student s[50];
int i,j,n;
clrscr();
printf("\n enter the limit");
```

```
scanf("%d",&n);
for(i=0; i< n; i++)
{ s[i].total=0;
printf("\n enter the roll no\n");
scanf("%d",&s[i].rollno);
printf("\n enter the name \n");
scanf("%s",s[i].name);
for(j=0;j<=3;j++)
{ printf("Enter the mark %d" , j+1);
scanf("%d",&s[i].marks[j])
s[i].total= s[i].total + s[i].marks[j];.
}
s[i].percent=s[i].total/3;
}
printf("STUDENT DETAILS ARE AS FOLLOWS:");
printf("\n_____________________________________");

for(i=0; i<n; i++)
{
printf("\n Roll No.: %d", s[i].rollno);
printf("\n Name : %s" , s[i].name);
if(s[i].percent > 75.0)
printf("\nStudent has Grade A quality");
else
if(s[i].percent > 60.0 && s[i].percent < 75.0)
printf("\nStudent has Grade B quality");
else
printf("\nStudent has Grade C quality");
printf("\n");
}
getch();
}
```

➲ 16. Write down a program in C to express any multiword string into its equivalent abbreviated form separated by a dot(.) i.e., SWAPAN KUMAR SEN will be written as S.K.S. **[WBSCTE 2016]**

Answer:

```
#include <stdio.h>
int main()
{
    char fname[20], mname[20], lname[20]; /* person's name */
     /* accept full name */
    printf("Enter full name (first middle last): ");
    scanf("%s %s %s", fname, mname, lname);
     /* print abbreviated name */
    printf("Abbreviated name: ");
    printf("%c. %c. %c\n", fname[0], mname[0], lname[0]);
    return 0;
}
```

Unit 4: Functions

Unit at a glance:

4.1 Definition and Significance

A function is a "black box" that we have locked part of our program into. The idea behind a function is that it compartmentalizes part of the program, and in particular, that the code within the function has some useful properties:

4.2 Recursion

When a function is called from the body of itself then the function is said to be as recursive function. Whenever function is called from itself then there is a chance that the function will enter into an infinite loop. To prevent a program from entering into an infinite loop in such a manner, every recursive function must have a terminating condition that decides whether to call the function once more or not. This terminating condition is also known as stopping rule in recursive function.

4.3 Storage Classes in C

To define fully a variable we need to mention:

1. The type of variable i.e. (int, float, char)
2. The storage class of variable.

There are two kinds of locations in a computer where such value may be kept and they are:

a) Memory and b) CPU registers.

Every C variable has a storage class and a scope. The storage class determines the part of memory where storage is allocated for an object and how long the storage allocation continues to exist. It also determines the scope, which specifies the part of the program over which a variable name is visible, i.e. the variable is accessible by name.

There are four types of storage classes available in 'C'.

1. Automatic storage class
2. Register storage class
3. External storage class
4. Static storage class

Automatic storage class:

Automatic variables are declared at the start of a block. Memory is allocated automatically upon entry to a block and freed automatically upon exit from the block.

Register Storage class:

Register variables are a special case of automatic variables. Variables which are used repeatedly or whose access times are critical may be declared to be of storage class register.

External Storage Class:

In some applications it may be useful to have data which is accessible from within any block and/or which remains in existence for the entire execution of the program. Such variables are called global variables, and the C language provides storage classes which can meet these requirements; namely, the external and static classes.

Static Storage class:

Like external variables, the storage class, static, similarly provides a lifetime over the entire program, however, it provides a way to limit the scope of such variables. Static storage class is declared with the keyword static when the variable is defined.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. An array is passed to a function as argument to the function in C language equivalent [WBSCTE 2007]

(a) call by value (b) call by reference
(c) call by name (d) none of these

Answer: (b)

2. Write the output of the following program fragment: [WBSCTE 2007, 2008]

```
main( )
{
   printf("Hello World");
   main( );
}
```

(a) compilation error (b) run time error
(c) infinite loop (d) none of these

Answer: (c)

(Recursively main( ) calls main( ) infinite times as there is no stopping rule.)

3. Find the output of the program: [WBSCTE 2007, 2009]

```
main( )
{
   printf("%d", f(8,4));
}
int f(int a, int b)
{
   if(a= =b)
         return a;
   else if(a>b)
         return f(a-b, b);
   else
         return f(a, b-a);
}
```

(a) 2 (b) 4 (c) 1 (d) none of these

Answer: (b)
(On first call of f( ), a>b is satisfied and recursively f(4, 4) is called. Now a==b is satisfied, and 4 is returned to calling function f( ), which returns 4 also to main( ).)

4. Find the output of the following programs:

```
main ( )
{printf("%d", f(8, 4)),}
int f(int a, int b) {if (a = = b) return a; else if (a > b)
return
f(a - b, b); else
return f(a, b-a);
```

[WBSCTE 2010]

(a) 2 (b) 4 (c) 0 (d) none of these

Answer: (b)

5. Array passed as argument to a function [WBSCTE 2010]
(a) Address of the array (b) value of the first eliment of the array
(c) Address of the first eliment of the array (d) No of elements of the array

Answer: (c)

6. When we mention the prototype of a function? [WBSCTE 2011]
(a) Defining (b) Declaring (c) Prototyping (d) Calling

Answer: (b)

7. How would you round off a value from 1.66 to 2.0? [WBSCTE 2011]
(a) ceil(1.66) (b) floor(1.66) (c) roundup(1.66) (d) roundto(1.66)

Answer: (a)

8. Which of the following statements should be used to obtain a remainder after dividing 3.14 by 2.1? [WBSCTE 2011]
(a) rem = 3.14%2.1;
(b) rem = modf(3.14, 2.1);
(c) rem = fmod(3.14, 2.1);
(d) Remainder cannot be obtained in floating point division

Answer: (c)

9. The default storage class of a local variable is [WBSCTE 2017]
(a) auto (b) static (c) register (d) extern

Answer: (a)

B. State whether the following statements are True or False:

10. The return type of a function is int by default. [WBSCTE 2008, 2009, 2010, 2014]
Answer: True

11. A function can be defined and placed before the main( ) function. [WBSCTE 2008]
Answer: True

12. The returns type of function is int by default – [WBSCTE 2009]

Answer: True

13. Too many recursive calls may result a stack overflow. **[WBSCTE 2010]**

Answer: True

(Each time a recursion function is called, it stores the calling address at the stack.)

14. All functions including 'main' can be called recursively. **[WBSCTE 2010]**

Answer: True

C. Answer the following questions:

15. Why we write void main( )? **[WBSCTE 2007]**

Answer:

void means that the function has no return-value, so we do not need return.

16. State the difference between Global and Local variable. **[WBSCTE 2009]**

Answer:

Difference between Global and Local Variable: Global variable can be accessed from anywhere from the program, but local variable is accessible only from the block, where local variable is declared.

17. Distinguish between –

(i) & operator and * operator **[WBSCTE 2009]**

(ii) Actual and Formal arguments **[WBSCTE 2009]**

(iii) Automatic and static variables **[WBSCTE 2010]**

(iv) Scope and Visibility of variables **[WBSCTE 2010]**

Answer:

(i) & denotes "address" of a variable, whereas

* denotes "content of an address".

(ii) *Difference between actual and formal arguments:*

Actual arguments are variables whose values are supplied to the function in any function call.

Formal arguments are variables used to receive the values of actual arguments from the calling program.

(iii) Automatic → Variable declaration is varied by program.

Static→ Variable declaration is fixed for a program.

(iv) Scope refers to where variable can be used in your program, where as Visibility refers to lifetime of variable. Though sense are different but meaning is same.

18. (i) What is prototyping? **[WBSCTE 2009]**

(ii)What is the meaning of prototype of a function? **[WBSCTE 2011]**

OR,

What is prototype declaration of a function? **[WBSCTE 2017]**

Answer:

(i) *Prototyping:* Prototyping is a declaration where the types of parameters are explicit and are part of the type of the function.

(ii) Declaration of function is known as prototype of a function. Prototype of a function means

(1) What is return type of function

(2) What parameters are we passing.

19. Write the output of the following program fragment: **[WBSCTE 2009]**

```
main( )
{
int i=5, j=2;
junk(i, j);
printf("\n%d%d", i, j);
}
junk(int i, int j)
{
i=i*i;
j=j*j;
}
```

Answer: 52 (Since, no value is returned from junk function).

20. Can you pass a function to other functions? If yes, how? **[WBSCTE 2010]**

Answer:

No.(The returned value of a function can be passed to another function, but a function itself cannot be passed to another.)

21. Write a C program to test if a given number is a power of 2. **[WBSCTE 2011]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#define CheckPowof2(a) ( a & (a-1))
void main()
{
clrscr();
int a;
printf("Enter the number to Check if it is power of 2 or not:");
scanf("%d", &a);
if ((CheckPowof2(a)) == 0 )
printf("\nNumber %d is indeed a power of 2\n ",a);
else
printf("\nNumber %d is not a power of 2\n ",a);
getch();
}
```

22. Describe prototype function declaration? **[WBSCTE 2015]**

Answer: A function prototype is a function declaration that specifies the data types of its arguments in the parameter list. The compiler uses the information in a function prototype to ensure that the corresponding function definition and all corresponding function declarations and calls within the scope of the prototype contain the correct number of arguments or parameters, and that each argument or parameter is of the correct data type.

23. What do you mean by recursion? **[WBSCTE 2015]**

Answer:

When a function is called from the body of itself then the function is said to be as recursive function.

24. What is the output of the code? **[WBSCTE 2017]**

a)
```
intfunc();
main()
{
printf("%d",func());
printf("%d",func());
printf("%d",func());
}
int func()
{
int counter=5;
counter++;
return(counter);
}
```

Answer: Output 6 6 6

b)
```
int f2(int a)
{
return(a*a);
}
int f1(int a, int b)
{
Return(f2(a*b));
}
main()
{
int x;
x=f1(2,4);
printf("%d",x);
}
```

Answer: Output: 64

Long Answer Type Questions

1. What are the disadvantages of using recursion in C? Use recursive function calls to evaluate $f(x) = x-x^3/3!+x^5/5!-x^7/7!+\ldots\ldots\ldots\ldots$ **[WBSCTE 2006, 2010]**

Answer:

Disadvantages of recursion:

(i) If the stopping rule is not correctly implemented, then the function may enter into initiate loop giving us an erroneous output.

(ii) The performance of recursion degrades for compilations, which are larger in breadth.

Evaluation of f(x):

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(int, int);
long fact(int);
void main( )
{
clrscr( );
int x, n;
float ans;
printf("Enter n:");        /****Input no. of terms****/
scanf("%d", &n);
printf("Enter x:");
scanf("%d", &x);
ans=f(x,n);
printf("%f", ans);
getch( );
}
float f(int x, int n)
{
if(n==0)                   /****stopping rule****/
{
return((float)x);
}
else
{
return ((pow(-1,n)*(pow(x,(2*n+1))/fact(2*n+1)))+f(x,n-1));
}
}
/****factorial of a number*****/
long fact(int n)
{
if (n == 0)
return 1;
else
return(n * fact(n-1));
}
```

2. Write a recursive function to display the first n-terms of a Fibonacci series. **[WBSCTE 2007, 2010]**

Answer:

```
#include<stdio.h>
#include<conio.h>
int fib(int);
int a=0,b=1,c;
void main( )
{
int n;
printf("Enter n:");
```

```
scanf("%d",&n);
printf("%d %d ",a,b);
printf("%d",fib(n));
getch( );
}
int fib(int m)
{
if(m==3)
return 1;
else
{
c=fib(m-1);
printf("%d ",c);
a=b;
b=c;
c=a+b;
return c;
}
}
```

3. Differentiate between call by value and call by reference. [WBSCTE 2007, 2010]

Answer:

Call by value	Call by address
(i) In this type of parameter passing the caller function passes a temporary copy of the parameter rather than the original one to the called function.	(i) In this type of parameter passing the caller function only allows to change the internal value of the in the called function.
(ii) Default mechanism of parameter passing is called called by value or passed by value.	(ii) It is another procedure of passing value or argument to a function.
(iii) Whenever we call a function and pass something to it we have always passed the values of variables to the called function.	(iii) In passing by reference or passing by address the address of the value is passed as an argument to the uses defined function that is why this address is relating the modified value and return to the main function.

4. When not to use recursion?—Discuss. **[WBSCTE 2008]**

Answer:

Recursion is the technique of defining anything in terms of itself. But not all problems have recursive solution. Recursion is generally a worse problem to go for simple problems, or problems not recursive in nature.

Where storage space is limited, we cannot use recursion, it consumes storage space. Recursion uses stack to implement a function. If the number of call of a recursive a function to itself is high enough, then there may be a stack overflow, which causes severe error.

5. What do you mean by recursion? Write a program to find the GCD of two numbers. Do you think the iteration method is better than recursion? **[WBSCTE 2008]**

Answer:

Recursion: When a function is called from the body of itself then the function is said to be as recursive function. Whenever function is called from itself then there is a change that the function will enter into an infinite loop in such a manner, every recursive function must have a terminating condition that decides whether to call the function once more or not. This terminating condition is also known as *stopping rule* in recursive function.

Program to find GCD:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
int x,y,r;
printf("Enter two numbers in ascending order:");
scanf("%d%d",&x,&y);
do
{
r=y%x;
if(r==0)
{
printf("G.C.D. of the two nos. is=%d", x);
}
y=x;
x=r;
}
while(r!=0);
}
```

In iteration method, the complexity and the size of the program increase. But, in case of recursion, if no. of recursive calls is high enough, or there is low memory space problem, then stack overflow may create a severe problem on execution of the program. So, both the methods have some advantages and disadvantages. Which one is better method that depends on situations.

➲ 6. Describe the utility of using 'function' through an example. **[WBSCTE 2008]**

Answer:

A function is a block of code that performs a calculation and returns a value. Functions are used to –

1. increase the readability of a program
2. to perform a task in repetitive manner

Suppose we want to calculate the 10th power of 2 and 4. If we do not use a function, the code will be longer enough than when we use a function to calculate 10th power.

```
/*Program to find power without using function*/
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
```

```
int x, n, i, ans=1;
n=10;
x=2;
for(i = 0; i < n; i++)
ans*= x;
printf("%d^%d=%d", x, n, ans);
x=4;                                    /*same code implemented
twice*/
ans=1;
for(i = 0; i < n; i++)
ans*= x;
printf("%d^%d=%d", x, n, ans);
getch( );
}
/*Program to find power using function*/
#include<stdio.h>
#include<conio.h>
int power(int, int);
void main( )
{
clrscr( );
int x, n;
n=10;
power(2,n);
power(4, n);
getch( );
}
int power(int x, int n)
{
int ans = 1, i;
for(i = 0; i < n; i++)
ans*= x;
printf("%d^%d=%d", x, n, ans);
return ans;
}
```

So, if we use functions, program size decreases, and the program becomes well structured.

7. State three advantages of using functions. What is the purpose of the return statement? What are the two principal components of a function definition? **[WBSCTE 2009]**

Answer:

The advantage of using functions in C are as follows .....

1. Debugging is easier.
2. It is easier to understand the logic involved in the program
3. Testing is easier.

Purpose of Return statement: The return statement terminates the execution of a function and returns control to the calling function. Execution resumes in the calling

function at the point immediately following the call. A return statement can also return a value to the calling function.

The principle components of function definition are – (1) function prototype, (2) function arguments.

8. The main is a user defined function. How does it differ from other user-defined functions? [WBSCTE 2009]

Answer:

main is defined as the entry point if you know how to design a compiler then you will be able to chage the entry point accordingly. you can definetly change the arguments passed to main anytime. These are all valid statement

void main()
int main(char p)
char main(int x,float y)
void main(int x,float y,double z) etc ...

This is the reason why main is called an user defined function.

There is no difference between user defined function and main function, main is just like any other user defined function but it is a compulsory function from where the execution starts.

9. What is the difference between Recursion and Iteration? Write a recursive function to generate the value of F(n) for a given n where F(n) is defined as: F(n) = F(n -1)/2 + 3 and F(0) =1. [WBSCTE 2010]

Answer:

A recursive function works through the process of calling itself until a condition is met. Iteration uses a looping control structure (while, do while, for) to repeat a section of code until a condition is met.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
float f(int);
void main ( )
{
clrscr();
int n;
float ans;
printf("Enter n:");
scanf("%d", &n);
ans=f(n);
printf("Answer:=%f", ans);
getch();
}
float f(int n)
{
if(n==0)
return 1;
else
```

```
return(3+f(n-1)/2);
}
```

10. Can function return more than one result? If yes, how? **[WBSCTE 2010]**

Answer:
A C function can return only one value. However, you can return multiple values by wrapping them in a class or struct.

```
struct Foo
{
    int value1;
    int value2;
};
Foo SomeFunction()
{
    Foo result = { 5, 4 };
    return result;
}
```

11. Explain the concept of making function calls. **[WBSCTE 2011]**

Answer:
Functions are used for Placing or Storing the Code which is to be Repeated Several Times. For Example, if we need Same Code, then we must have to Write that Code Again and Again So that for Removing this Task, we uses functions.

There are Two Ways for Calling a Function

1) Call by Value: when we call a Function and if a function can accept the Arguments from the Called Function, Then we must have to Supply some Arguments to the Function. So that the Arguments those are passed to that function just contains the values from the variables but not an Actual Address of the variable. So that generally when we call a Function then we will just pass the variables or the Arguments and we doesn't Pass the Address of Variables , So that the function will never effects on the Values or on the variables. So Call by value is just the Concept in which you must have to Remember that the values those are Passed to the Functions will never effect the Actual Values those are Stored into the variables.

2) Call By Reference : When a function is called by the reference then the values those are passed in the calling functions are affected when they are passed by Reference Means they change their value when they passed by the References. In the Call by Reference we pass the Address of the variables whose Arguments are also Send. So that when we use the Reference then, we pass the Address the Variables. When we pass the Address of variables to the Arguments then a Function may effect on the Variables. Means When a Function will Change the Values then the values of Variables gets Automatically Changed. And When a Function performs Some Operation on the Passed values, then this will also effect on the Actual Values.

12. Write a C program using function to calculate x to the power of y, where y can be either negative or positive. [WBSCTE 2011]

OR,

Two numbers are entered through keyboard. Write a program to find the value of one number raised to the power of another. [WBSCTE 2017]

Answer:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
double power(double,double);
void main()
{
clrscr();
double x,y;
printf("Enter the value of x");
scanf("%lf",&x);
printf("Enter the value of y");
scanf("%lf",&y);
printf("The value of x to the power of y is %lf",power(x,y));
getch();
}
double power(double x,double y)
{
int flag=0;
double i,c=1.0;
if(y<0)
{
flag=1;
y*=-1;
}
for (i=1.0;i<=y;i++)
c=c*x;
if(flag==1)
return(1/c);
else
return(c);
}
```

13. Predict the output or error(s) for the following and explain. [WBSCTE 2011]

Answer:

```
main ( )
{
int i=1;
while (i<=5)
{
printf("%d",i);
if (i>2)
goto here;
i++;
```

}

Output	Explanation
Label here used but not defined	The label of the goto statement should be defined some where in the function, otherwise the syntax error message will be displayed.

14. Write a C program to find out a string consists of how many consonants or vowels or digits or special characters. **[WBSCTE 2011]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
char s[100];
int b,c,d,e,i,l,p;
b=c=d=e=0;
clrscr();
printf("\n\t Enter the string :=>");
gets(s);
l=strlen(s);
for(i=0;s[i]!='\0';i++)
{
if((s[i]>=65&&s[i]<=90||s[i]>=97&&s[i]<=122))
b++;
if(s[i]>=48&&s[i]<=57)
c++;
if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u'||s[i]=='
A'||s[i]=='E'||s[i]=='I'||s[i]=='O'||s[i]=='U')
d++;
if (s[i]==' ')
e++;
}
p=b-d;
printf("\n\t No of vowels=%d \n",d);
printf("\n\t No of consonants=%d \n ",p);
printf("\n\t No of digits=%d \n",c);
printf("\n\t no of words=%d \n ",e+1);
printf("\n\t No of specal character=%d \n",l-b-c-e);
getch();
}
```

15. Predict the output or error(s) for the following and explain. **[WBSCTE 2011]**

Answer:

```
main()
      {
int i=();
for (;i++;);
printf("%d",i);
```

```
}
```

Output	Explanation
1	Since the loop runs once and the value of i increases by 1 due to i++, so the value of i outside the loop is 1.

16. Write a program to read and print an array of n numbers using function. **[WBSCTE 2011]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
void arrayprint(int*arr,int);
void main()
{
clrscr();
int *arr,i,n;
printf("Enter the value of n:");
scanf("%d",&n);
arr=(int*)malloc(n*sizeof(int));
for (i=0;i<n;i++)
{
printf("Enter the number:");
scanf("%d",&arr[i]);
}
arrayprint(arr,n);
getch();
}
void arrayprint(int*arr,int n)
{
int i;
printf("The array is:\n");
for(i=0;i<n;i++)
printf("%d\n",arr[i]);
}
```

17. What is the difference between pass by value and pass by reference OR pass by address in C? Explain with example. **[WBSCTE 2011]**

Answer:

The arguments passed to function can be of two types namely

1. Values passed 2. Address passed

The first type refers to call by value and the second type refers to call by reference.

For instance consider program1

```
main()
{
int x=50, y=70;
interchange(x,y);
printf("x=%d y=%d",x,y);
```

```
}
interchange(x1,y1)
int x1,y1;
{
int z1;
z1=x1;
x1=y1;
y1=z1;
printf("x1=%d y1=%d",x1,y1);
}
```

Here the value to function interchange is passed by value.

Consider program2

```
main()
{
int x=50, y=70;
interchange(&x,&y);
printf("x=%d y=%d",x,y);
}
interchange(x1,y1)
int *x1,*y1;
{
int z1;
z1=*x1;
*x1=*y1;
*y1=z1;
printf("*x=%d *y=%d",x1,y1);
}
```

Here the function is called by reference. In other words address is passed by using symbol & and the value is accessed by using symbol *.

The main difference between them can be seen by analyzing the output of program1 and program2.

The output of program1 that is call by value is

x1=70 y1=50

x=50 y=70

But the output of program2 that is call by reference is

*x=70 *y=50

x=70 y=50

This is because in case of call by value the value is passed to function named as interchange and there the value got interchanged and got printed as

x1=70 y1=50

and again since no values are returned back and therefore original values of x and y as in main function namely

x=50 y=70 got printed.

But in case of call by reference address of the variable got passed and therefore what ever changes that happened in function interchange got reflected in the address location and therefore the got reflected in original function call in main also without explicit return value. So value got printed as *x=70 *y=50 and x=70 y=50

18. (a) What do you mean by parameter passing in C?
(b) Discuss different types of parameter passing supported by C through appropriate examples. **[WBSCTE 2015, 2016]**

Answer:

a) Passing parameter to Function :

- In C Programming we have different ways of parameter passing schemes such as Call by Value and Call by Reference.
- Function is good programming style in which we can write reusable code that can be called whenever require.
- Whenever we call a function then sequence of executable statements gets executed. We can pass some of the information to the function for processing called argument.

b) Two Ways of Passing Argument to Function in C Language :

- Call by Reference
- Call by Value

Let us discuss different ways one by one –

A.Call by Value :

```
#include<stdio.h>
void interchange(int number1,int number2)
{
int temp;
temp = number1;
number1 = number2;
number2 = temp;
}
int main() {
int num1=50,num2=70;
interchange(num1,num2);
printf("\nNumber 1 : %d",num1);
printf("\nNumber 2 : %d",num2);
return(0);
}
```

Output :

Number 1 : 50

Number 2 : 70

Explanation : Call by Value

- While Passing Parameters using call by value , xerox copy of original parameter is created and passed to the called function.
- Any update made inside method will not affect the original value of variable in calling function.
- In the above example num1 and num2 are the original values and xerox copy of these values is passed to the function and these values are copied into number1,number2 variable of sum function respectively.

- As their scope is limited to only function so they cannot alter the values inside main function.

B.Call by Reference/Pointer/Address:

```
#include<stdio.h>
void interchange(int *num1,int *num2)
{
int temp;
temp  = *num1;
*num1 = *num2;
*num2 = temp;
}
int main() {
int num1=50,num2=70;
interchange(&num1,&num2);
printf("\nNumber 1 : %d",num1);
printf("\nNumber 2 : %d",num2);
return(0);
}
```

Output .

Number 1 : 70

Number 2 : 50

Explanation : Call by Address

- While passing parameter using call by address scheme , we are passing the actual address of the variable to the called function.
- Any updates made inside the called function will modify the original copy since we are directly modifying the content of the exact memory location.

19. a) Differentiate between formal parameters and actual parameters.

b) Write a recursive function to get GCD of two numbers. **[WBSCTE 2017]**

Answer:

(a) Actual arguments:

The arguments that are passed in a function call are called actual arguments. These arguments are defined in the calling function.

Formal arguments:

The formal arguments are the parameters/arguments in a function declaration. The scope of formal arguments is local to the function definition in which they are used. Formal arguments belong to the called function. Formal arguments are a copy of the actual arguments. A change in formal arguments would not be reflected in the actual arguments.

```
int fn(int a)
 {
 int square;
square=a*a;
 return square;
 }
```

```
 void main()
{
 int A=2;
 int sq;
 sq=fn(A);
 }
```

In the above example 'A' is actual parameter and 'a' is formal parameter

b)

```
#include <stdio.h>
int gcd (int n1, int n2);
int main()
{
   int n1, n2;
   printf("Enter two positive integers in ascending order: ");
   scanf("%d %d", &n1, &n2);
   printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1,n2));
   return 0;
}
int gcd (int n1, int n2)
{
   if (n2 != 0)
      return hcf(n2, n1%n2);
   else
      return n1;
}
```

Unit 5: Pointers

Unit at a glance:

5.1 Pointer declaration and Initialization

The first thing to do with pointers is to declare a pointer variable, then set it to point somewhere, and finally manipulate the value that it points to. A simple pointer declaration looks like this:

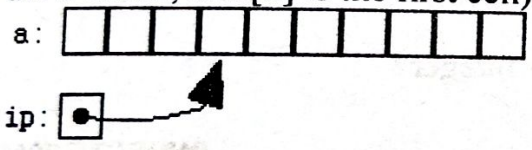
```
int * ptr1 ; /* declares an integer pointer*/
float * ptr2; /* declares an float pointer*/
char * ptr3 ; /* declares an character pointer*/
```

5.2 Pointers and Arrays

Pointers can also point at the cells of an array. For example, we can write

```
int *ip;
int a[10];
ip = &a[3];
```

and we would end up with ip pointing at the fourth cell of the array a (remember, arrays are 0-based, so a[0] is the first cell). We could illustrate the situation like this:



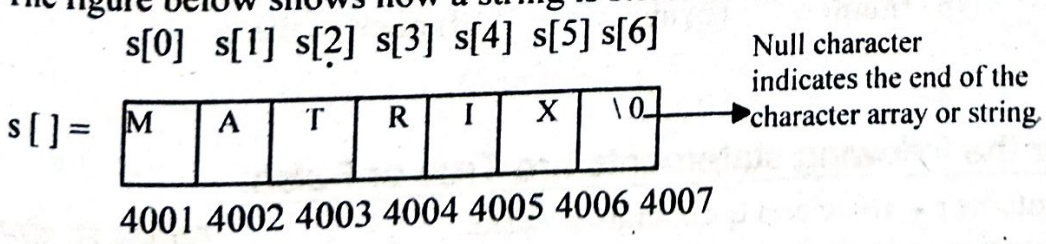
*ip gives us what ip points to, which in this case will be the value in a[3].
Once we have a pointer pointing into an array, we can start doing *pointer arithmetic*. Given that ip is a pointer to a[3], we can add 1 to ip:

```
ip + 1
```

5.3 Pointer and Strings

In C, character arrays are often called strings. A string in C is always terminated by a null character ('\0'). The ASCII value of '\0' is 0. The figure below shows how a string is stored and accessed.

The figure below shows how a string is stored and accessed.



Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. If P1 and P2 are properly declared and initialized pointers and x,y,z are integer variables, then which of the following statement is not valid? [WBSCTE 2007]
   (a) x = x + *P1 (b) y = *P1 * *P2 (c) Z = Z*-P2/*P1 (d) *P2 = *P2 + 10

Answer: (c)
(P2/*P1 is not possible)

2. If P and Q are properly declared and initialized pointers and x, y and z are integer variables, then which of the following statement is not valid? [WBSCTE 2007]
   (a) x=x+*P (b) y=*P+*Q (c) P=Q+10 (d) *P=P+10

Answer: (d)

3. Can you combine the following two statements into one?
   char *p
   p = (char*) malloc(100);
   [WBSCTE 2011]
   (a) char p = *malloc(100) (b) char *p = (char) malloc(100)
   (c) char *p = (char*) malloc(100) (d) char *p = (char*) (malloc*) (100)

Answer: (c)

4. In the following code, the P2 is Integer Pointer or Integer?
   typederf int*ptr;
   ptr p1, p2;
   [WBSCTE 2011]
   (a) Integer (b) Integer pointer
   (c) Error in declaration (d) None of above

Answer: (b)

5. **Can you combine the following two statements into one?** [WBSCTE 2012, 2013]
   char *p;
   p = (char*)malloc(100);
   (a) char p =*malloc(100) (b) char *p = (char) malloc(100)
   (c) char *p = (char*)malloc(100) (d) char *p = (char*)(malloc*)(100)

Answer: (c)

6. *(&num) is equivalent to writing
   [WBSCTE 2017]
   (a) &num (b) *num (c) num (d) none of these

Answer: (c)

B. State whether the following statements are True or False:

7. int *p, q; p –contains r – value and q-contains I-value. [WBSCTE 2007]

Answer: False (p contains address)

8. Pointers can be used to access the members of structure variables. [WBSCTE 2008]
Answer: True
(Using strct →member_name, we can access members of a structure using pointer.)

9. The underlying type of a pointer variable is void. [WBSCTE 2008]
Answer: True

10. (x[2][5] and *(*(x+2)+5) are not same, and also x[2][5] means element of 3rd row and 6th column.)
"%p" is a valid format string in C. [WBSCTE 2010]
Answer: False

C. Answer the following questions:

11. What is null pointer? [WBSCTE 2008, 2011, 2015,2016]
Answer:
A *null pointer* is a special pointer that does not point anywhere.

12. What is void pointer? [WBSCTE 2008]
Answer:
A void pointer is pointer which has no specified data type. The compiler has no idea what type of object a void pointer really points to.

13. State what happens after the execution of the following statements— [WBSCTE 2008]

(a) &a[0] = = a;
Answer:
&a[0] = = a is true; as both gives the base address of array a

(b) &a[i] = = a+i;
Answer:
& a [i] = = a + i is true & a [i] gives the address of i-th element of array a. Also (a+ i) gives the same address.

(c) int a, *b = &a;
Answer:
a is an integer variable. b is integer pointer. b keeps the address of a.

(d) char *s;
Answer:
s is a character type pointer; i.e., s keep the address of a character variable.

14. Given the following declarations: [WBSCTE 2009]

```
int x =10, y=10;
int *P1=&x, *P2=&y;
```

Want is the value of the following expression? (*P1)++
Answer:
The value of (*P1)++ is 11.

15. Distinguish between the following pair: **[WBSCTE 2009]**

(*m)[5] and *m[5].

Answer:

(*m)[5] → it is array of pointers

*m[5] → it is pointer of an array.

16. What does the following declaration mean?— **[WBSCTE 2010]**

```
int (*ptr)[10];
```

Answer:

The declaration means that *ptr is an array of pointers of integer type.

17. Predict are output or error(s) for the following and explain. **[WBSCTE 2011]**

```
main()
{
char*p;
printf("%d%d",sizeof(*p), sizeof(p));
}
```

Answer:

Output: 1 2

Explanation: The sizeof() operator gives the number of bytes taken by its operand. P is a character pointer, which needs one byte for storing its value (a character). Hence sizeof(*p) gives a value of 1. Since it needs two bytes to store the address of the character pointer sizeof(p) gives 2.

18. What is dangling pointer in C? **[WBSCTE 2011]**

Answer:

If any pointer is pointing the memory address of any variable but after some variable has deleted from that memory location while pointer is still pointing such memory location. Such pointer is known as dangling pointer.

19. What are the arithmetic operators that are permitted on pointers? **[WBSCTE 2011]**

Answer:

Addition and subtraction.

But if you cast the pointer to an arithmetic type with the appropriate size then you can do all sorts of arithmetic operations on it.

20. What is dangling pointer in C? **[WBSCTE 2011]**

Answer:

If any pointer is pointing the memory address of any variable but after some variable has deleted from that memory location while pointer is still pointing such memory location. Such pointer is known as dangling pointer.

21. Distinguish between malloc() and calloc() function. **[WBSCTE 2016]**

Answer:

Here are two major differences between malloc and calloc in C programming language: first, in the number of arguments. The malloc() takes a single argument, while calloc() takess two. Second, malloc() does not initialize the memory allocated, while calloc() initializes the allocated memory to ZERO.

22. What do you mean by pointer in C? **[WBSCTE 2016]**

Answer: ***Refer to article 5.1.***

23. How are arrays related to pointers? **[WBSCTE 2017]**

Answer:

Arrays are closely related to pointers in C programming but the important difference between them is that, a pointer variable takes different addresses as value whereas, in case of array it is fixed i.e. **Array name** is itself a pointer (constant pointer) which is pointing to the base address of the array, we cannot change the location it points at.

24. What is the output of the code? **[WBSCTE 2017]**

```
main()
{
      int arr[]={1,2,3,4,5};
      int *ptr,i;
      ptr=arr+4;
      for(i=0;i<5;i++)
            printf("%d",*(ptr-i));
}
```

Answer: Output: 5 4 3 2 1

Long Answer Type Questions

1. Write a C program which will count the number of vowels, consonants and other characters in a given string in a character pointer. **[WBSCTE 2007]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
voidscan_line(char line[], int *pv, int *pc, int *pd, int *pw,
int *po);
void main( )
{
clrscr( );
char line[80];
int vowels = 0;
int consonants = 0;
int digits = 0;
int whitespaces = 0;
int other = 0;
printf("Enter a line:");
gets(line);
scan_line(line, &vowels, &consonants, &digits, &whitespaces,
&other);
printf("\nNo. of vowels : %d", vowels);
printf("\nNo. of consonants : %d", consonants);
printf("\nNo. of digits : %d", digits);
```

```
printf("\nNo. of whitespaces : %d", whitespaces);
printf("\nNo. of other : %d", other);
getch( );
}
voidscan_line(char line[], int *pv, int *pc, int *pd, int *pw,
int *po)
{
char c;
int count = 0;
while((c=toupper(line[count]))!='\0')
{
if(c=='A'||c=='E'||c=='I'||c=='O'||c=='U')
++*pv;
else if(c>='A' && c<='Z')
++*pc;
else if(c>='0' && c<='9')
++*pd;
else if(c=='  '||c=='\t')
++*pw;
else
++*po;
count++;
}
return;
}
```

2. What is an array of pointers? How is it declared? Distinguish between the address stored in the pointer and the value of that address.

[WBSCTE 2008, 2009, 2011]

Answer:

An array of pointers is similar to an array of any predefined data type. As we know that a pointer variable always contains an address, an array of pointers is a collection of address. These can be addresses of ordinary isolated variables or of array elements.

The elements of an array of pointers are stored in the memory just like the elements of any other kind of array. All rules that apply to other arrays also apply to the array of pointers.

Pointer is a special type of variable which hold the address of same or other variables. Pointer is declared anywhere in the memory.

3. What will be output of following c program?

[WBSCTE 2011]

```
int *call();
void main(){
int *ptr;
ptr=call();
clrscr();
printf("%d",*ptr);
}
int * call(){
int x=25;
```

```
++x;
return &x;
}
```

Output: Garbage value

Explanation: variable x is local variable. Its scope and lifetime is within the function call hence after returning address of x variable x became dead and pointer is still pointing ptr is still pointing to that location.

Solution of this problem: Make the variable x is as static variable.

In other word we can say a pointer whose pointing object has been deleted is called dangling pointer.

4. What is wild pointer in c? **[WBSCTE 2011]**

Answer

A pointer in c which has not been initialized is known as wild pointer.

Example:

What will be output of following c program?

```
void main(){
int *ptr;
printf("%u\n",ptr);
printf("%d",*ptr);
}
```

Output: Any address

Garbage value

Here ptr is wild pointer because it has not been initialized.

There is difference between the NULL pointer and wild pointer. Null pointer points the base address of segmentwhile wild pointer doesn't point any specific memory location.

5. Predict the output or error(s) for the following and explain: **[WBSCTE 2011]**

```
main( )
{
char s[ ]= "man";
int i;
for(i=0;s[ i ];i++)
printf("\n%c%c%c%c",s[ i ],*(s+i),*(i+s),i[s]);
}
```

Answer:

Output	Explanation
mmmm aaaa nnnn	First line is blank because of \n in the formatted output string and the loop runs for 4 times as the length of the word is 3 plus 1 for the null character '\0'.

6. Write a C program to find out the number of occurrence of a given character in a String. **[WBSCTE 2012]**

Answer:

```
#include <ctype.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
size_t bytewise_pop_count(
unsigned char*bp,size_tl
{
if ( (bp) && (l) ){
return bytewise_pop_count(bp+1,l-1)+(bp[0]?1:0;
}
Return 0;
}
void mercilessly_complement_bytes(
unsigned char * bp, size_t l
{
/*
transform
0 -> 1
!0 -> 0
*/
if ( (bp) && (l) ) {
bp[0] = bp[0] ? 0 : 1;
mercilessly_complement_bytes(bp+1,l-1);
}
}
void xor_bytes(
unsigned char * bp1, unsigned char * bp2, size_t l
{
/* stores result in bp2 */
if ((bp1)&&(bp2)&& (l)) {
bp2[0] ^= bp1[0];
xor_bytes(bp1+1, bp2+1, l-1);
}
}
int main(int argc, char * * argv) {
char c;
size_t count;
size_t l;
char * string;
char * t;
if (argc < 3) {
fprintf(stderr, "\n" "==> not enough arguments -- need
char and string\n" "\n");
return EXIT_FAILURE;
}
c = argv[1][0];
string = argv[2];
if ( l = strlen(string) ) {
t = malloc(l);
memset(t, c, l);
xor_bytes(string, t, l);
mercilessly_complement_bytes(t, l);
```

```
count = bytewise_pop_count(t, 1);
free(t);
} else {
count = 0;
}
if ( isprint(c) ) {
printf("\n""==> occurences of char ``%c'' in string
``%s'': %zu\n" "\n", c, string? string: "<NULL>",count);
} else {
printf("\n""==> occurences of char ``%hhu'' in string
``%s'':%zu\n" "\n",c, string? string:"<NULL>",count);
}
return EXIT_SUCCESS;
}
```

7. Predict the output or error(s) for the following: Explain. **[WBSCTE 2012]**

```
main()
{
int a[ ]={1,2,3,4,5,6,7};
printf("%d",*(a+3)-*(a+1));
}
```

Answer:

*a = 1
*(a+3)= 4
*(a+1) = 2
*(a+3) - *(a+1) = 2

Answer is 2

8. (a) Explain the use of pointer
(b) What are the arithmetic operators that are permitted on pointers?
[WBSCTE 2013]

Answer:

a) Every variable is a memory location and every memory location has its address defined which can be accessed using ampersand (&) operator, which denotes an address in memory. A **pointer** is a variable whose value is the address of another variable, i.e., direct address of the memory location.

```
int    *ip;     /* pointer to an integer */
double *dp;     /* pointer to a double */
float  *fp;     /* pointer to a float */
char   *ch      /* pointer to a character */
A  sample program shows the use of pointers.
#include <stdio.h>
int main ()
{
int  var = 20;    /* actual variable declaration */
int  *ip;         /* pointer variable declaration */
ip = &var;  /* store address of var in pointer variable*/
printf("Address of var variable: %x\n", &var  );
/* address stored in pointer variable */
```

```
printf("Address stored in ip variable: %x\n", ip );
/* access the value using the pointer */
printf("Value of *ip variable: %d\n", *ip );
return 0;
}
```

b) Incrementing a Pointer

```
int *ptr, var[2]={20,29};
ptr=&var;
ptr++;// will denote ptr[1]=29
```

Decrementing operators:

```
ptr--;// now denotes ptr[0]=20
```

Comparison operators

Pointers may be compared by using relational operators, such as ==, <, and >. If p1 and p2 point to variables that are related to each other, such as elements of the same array, then p1 and p2 can be meaningfully compared

```
Int *ptr1,*ptr2,var={22,88};
Prt1=&var[0];
Ptr2=var;
If(ptr1==ptr2){
Printf("Equal");
}
```

9. What do you mean by dynamic memory allocation? How this can be implemented in C. **[WBSCTE 2015, 2016, 2017]**

Answer:

1st part:

Dynamic memory allocation is when an executing program requests that the operating system give it a block of main memory. The program then uses this memory for some purpose. Usually the purpose is to add a node to a data structure. In object oriented languages, dynamic memory allocation is used to get the memory for a new object.

The memory comes from above the static part of the data segment. Programs may request memory and may also return previously dynamically allocated memory. Memory may be returned whenever it is no longer needed. Memory can be returned in any order without any relation to the order in which it was allocated.

2nd part:

For example to allocate a memory for a integer variable is as follows

```
int *p;
P=(int *)malloc(2);
```

Number of bytes

In the above example 2 byte memory is allocated for storing a integer value and the starting address of the allocated memory will be sorted in pointer p.(int *) specifies the type of pointer in which address is sorted.

This allocated memory can be deallocated or given back to the system using free library function.

free(p)

The above statement will free the memory pointed by pointer p.
These library functions are defined in alloc.h header file. So to use these functions in the program alloc.h header file has to be included in the program.

10. (a) Create a one dimensional integer array. Now insert n integers randomly into it. Now find the greatest integer from the array using a pointer.
(b) What are the drawbacks of pointer? **[WBSCTE 2015]**

Answer:

a)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10],n,i,max;
int *p;
clrscr();
printf("Enter the size of array: ");
scanf("%d",&n);
printf("Enter %d elements in the array:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Elements in the array are:\n");

for(i=0;i<n;i++)
printf("%5d",a[i]);
p=&a[0];
max=a[0];
for(i=0;i<n;i++)
{
if(max<=*p)
max=*p;
p++;
}
printf("\nMaximum element in the array is: %d",max);
getch();
}
```

b) Drawbacks of pointers in c:

- Uninitialized pointers might cause segmentation fault.
- Dynamically allocated block needs to be freed explicitly. Otherwise, it would lead to memory leak.
- Pointers are slower than normal variables.
- If pointers are updated with incorrect values, it might lead to memory corruption.

11. a) What are the advantages of pointer variable over memory variable? **[WBSCTE 2016]**

OR,

Write the advantages of using pointers. **[WBSCTE 2017]**

b) Write a program in C to access the elements of a one dimensional array using pointer. **[WBSCTE 2016]**

Answer:

a) By using pointers we can access a data which is available outside a function. By using pointer variable we can implement dynamic memory allocation. With the help of pointer we can handle any type of data structure. Pointers can increase the execution speed of program.

b)

```
#include <stdio.h>
int main()
{
   int data[5], i;
   printf("Enter elements: ");
   for(i = 0; i < 5; ++i)
     scanf("%d", data + i);
   printf("You entered: \n");
   for(i = 0; i < 5; ++i)
     printf("%d\n", *(data + i));
   return 0;
}
```

12. a) Write C programs to implement i) strcpy(), ii) strlen() library function by yourself.

b) What do you mean by null character? Describe its uses in string manipulation. **[WBSCTE 2016]**

Answer:

a) The strlen() function counts the number of characters in a given string. The format of function is:

```
strlen(char *string);
```

The function strlen() is used to count the length of the string. The function strlen() will not consider the null symbol, for example, if we give the string 'Hallo world!\0' as input to strlen(), the value returned will be 12.

The strcpy() function copies the contents of one string into another.

The syntax of strcpy() is:

```
strcpy(char *s2, char *s1)
```

where s1 is the source string, s2 is the destination string and s1 is copied to s2.

strcpy() example

```
#include <stdio.h>
#include <string.h>
int main()
{
   char str1[10]= "awesome";
   char str2[10];
   char str3[10];
   strcpy(str2, str1);
   strcpy(str3, "well");
   puts(str2);
```

```
    puts(str3);
    return 0;
}
```

Output

```
awesome
well
```

strlen() example

```
#include <stdio.h>
#include <string.h>
int main()
{
    char a[20]="Program";
    char b[20]={'P','r','o','g','r','a','m','\0'};
    char c[20];
    printf("Enter string: ");
    gets(c);
    printf("Length of string a = %d \n",strlen(a));
    //calculates the length of string before null charcter.
    printf("Length of string b = %d \n",strlen(b));
    printf("Length of string c = %d \n",strlen(c));
    return 0;
}
```

Output

```
Enter string: String
Length of string a = 7
Length of string b = 7
Length of string c = 6
```

b) "Storing Similar Data Items," a string is a character array terminated by a null character (\()).

For instance, s character array, array_ch, declared in the following statement, is considered a character string:

```
char array_ch[7] = {'H', 'e', '1', '1', 'o', '!', '\0'};
```

In C, the null character can be used to mark the end of a string, or to return logical FALSE. C treats \0 as one character. Each character in a string takes only 1 byte.

A series of characters enclosed in double quotes("") is called a string constant. The C compiler can automatically add a null character(\0) at the end of a string constant to indicate the end of the string.

For example, the character string "A character string." Is considered a string constant; so is "Hello!".

13. a) Create a two dimensional integer array to insert some integers randomly into it. Now find the greatest and smallest integer from the array and also find their corresponding locations.

b) What do you mean by pointer to a function? Give an example.

[WBSCTE 2016]

Answer:

a) ***Refer to Question No. 10.a) of Long Answer Type Questions.***

b) A pointer to a function can be defined to hold the starting address of a function and the same can be used to invoke a function. It is also possible to pass addresses of different functions at different times thus making the function more flexible and abstract. The syntax of defining a pointer to a function is shown in Fig. 1.

pointer to function

```
ReturnType(*PtrToFn)(arguments_if_any);
```

Fig: 1 Syntax of defining a pointer to a function

The definition of a pointer to a function requires the function's return type and the function's argument list to be specified along with the pointer variable. It should be remembered that the function prototype or definition should be known before its address is assigned to a pointer.

Following is a simple example that shows declaration and function call using function pointer.

```
#include <stdio.h>
// A normal function with an int parameter
// and void return type
void fun(int a)
{
    printf("Value of a is %d\n", a);
}
int main()
{
    // fun_ptr is a pointer to function fun()
    void (*fun_ptr)(int) = &fun;

    /* The above line is equivalent of following two
       void (*fun_ptr)(int);
       fun_ptr = &fun;
    */
    // Invoking fun() using fun_ptr
    (*fun_ptr)(10);
    return 0;
}
```

Output:

Value of a is 10

Unit 6: Structure & Unions and Enumerated Data Type

Unit at a glance:

6.1 Defining a Structure

The basic user-defined data type in C is the structure, or struct. (C structures are analogous to the records found in some other languages.) Defining structures is a two-step process: first you define a "template" which describes the new type, then you declare variables having the new type (or functions re-turning the new type, etc.). As a simple example, suppose we wanted to define our own type for representing complex numbers. (a complex number consists of a "real" and "imaginary" part, where the imaginary part is some multiple of the square root of negative 1.) FORTRAN has a built-in complex type, but C does not.

6.2 Union

A union is a variable that may hold (at different times) objects of different types and sizes, with the compiler keeping track of size and alignment requirements. Unions provide a way to manipulate different kinds of data in a single area of storage, without embedding any machine-dependent information in the program.

Union looks like a structure but it is not a structure. A union is defined as given below:

```
union union_name
{
data_type_1 member1 ;
data_type_2 member2 ;
data_type_3 member3 ;
_____________________
_____________________
data_type_k memberk ;
};
```

where memberk is the kth member of the union and data_typed_k is the type of data which will be assigned to member k. We can define a variable of this union type in the following way: union union_name var_name

The definition and declaration of unions are exactly as we had done in case of structure. But there is important difference. The variable of this union type can, at any point of time store, any one of the members. When the variable is defined, the compiler reserves memory for storing any one of the members. Most likely, the members will require different number of bytes for their storage. That is why the compiler reserves enough bytes so that the member with the largest memory requirement can be stored.

Example: 1

```
struct example
{
int x;
float y;
char z;
};
struct example m;
```

Example: 2

```
union example
{
int x;
float y;
char z;
};
union example n;
```

Now, in case of the structure variable m, compiler will allocate 7 bytes (2 bytes for integer, 4 bytes for float and 1 byte for char) whereas in the case of union variable n, compiler will allocate 4 bytes, so that any member of that union can be stored.

6.3 Enumerated Data Type

The enumerated data types are also the user defined data types but they are somewhat different, from the typedef data types. Enumerated data type provides a way for attaching names to numbers, thereby increasing comprehensibility of the code. These data types work when you know in advance a finite list of values that a data type can take on. For example, if you want to define a data type for the days of a week, you know in advance the days of week can have seven values, i.e., Sunday, Monday, ... Saturday. Now, we can define an enumerated data type for the above given days. Note: The variable defined for the enumerated date type can access only the member of that data type, it canot access or assign any other value. The general format for the enumerated data type is: enum user_define_name {element_1,
element_2,. . . . . .element_n} ; In the above syntax, enum is a keyword telling the compiler about the declaration of enumerated data type.
user_define_name is the name given to the enumerated data type.
For the week, we can define the enumerated data type as follows:
enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}; week obj1;

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. The typedef statement is used for – [WBSCTE 2007]
   (a) declaring user defined data types
   (b) declaring variant variables
   (c) for type casting of variables
   (d) none of the above

Answer: (d)
(typedef is used to assign alternative names to existing types, such as structure and union.)

2. The member variables of structure are accessed by using – **[WBSCTE 2007]**
(a) dot (.) operator
(b) arrow (→) operator
(c) asterisk (*) operator
(d) ampersand (&) operator

Answer: (a)

3. Identify the most appropriate sentence to describe union – **[WBSCTE 2007]**
(a) unions contain members of different data types witch share the same storage area in memory
(b) unions are like structures
(c) unions are less frequency used in the program
(d) unions are used for self operations.

Answer: (a)
unions contain members of different data types which share the same storage area in memory.

4. The union holds – **[WBSCTE 2007, 2010]**
(a) one object at a time
(b) multiple objects
(c) both (a) and (b)
(d) none of these

Answer: (a)
(structure holds all objects at a time, but union can hold one object at a time.)

5. Bit fields are used only with – **[WBSCTE 2007]**
(a) unsigned int data type
(b) float data type
(c) char data type
(d) int data type

Answer: (a)

6. The number of bytes required for enumerated data type in memory are – **[WBSCTE 2007]**
(a) 2 bytes (b) 4 bytes (c) 1 byte (d) 3 bytes

Answer: (a)
(Because, it actually stores integer type of data.)

7. What is the similarity between a structure, union and enumeration? **[WBSCTE 2011]**
(a) All of them let you define new values
(b) All of them let you define new data types
(c) All of them let you define new pointers
(d) All of them let you define new structures

Answer: (b)

8. The variables declared in a structure definition are called its **[WBSCTE 2015]**
(a) group variables
(b) merged variables
(c) member variables
(d) total variables

Answer: (c)

9. A structure can be placed within another structure is known as [WBSCTE 2017]
(a) nested structure (b) self-referential structure
(c) parallel structure (d) pointer to structure

Answer: (a)

B. Fill in the blanks in the following statements:

10. The key word ***typedef*** can be used to create a data type identifier. [WBSCTE 2009]

C. State whether the following statements are True or False:

11. All structure elements are stored in contiguous locations – [WBSCTE 2005]

Answer: True

(Structures stores elements like array. The only difference is array stores similar type of elements, but structure can store different types of elements.)

12. The members within a union share the same storage area – [WBSCTE 2005]

Answer: True

D. Answer the following questions:

13. State the difference between a structure and an array. [WBSCTE 2008]

Answer:

A structure is a collection of different types of data. An array is a continuous memory location of homogenous data type.

14. What keyword is used in C to create a structure? [WBSCTE 2008]

Answer: In C to create a structure, the keyword "struct" is used.

15. Write code that defines a structure named time, which contains three int members. [WBSCTE 2008]

Answer:

```
struct time
{
int t;
int m;
int s;
};
```

16. State the difference between structure and union. [WBSCTE 2008, 2009, 2017]

Answer:

Difference between structure and union: In structures, each member has its own storage location, whereas all the members of a union use the same location. This implies that, although a union may contain many members of different types, it can handle only one member at a time.

17. State one application for which structure is used. [WBSCTE 2009]

Answer: Structure is used for linked list implementation.

18. What are the enumeration variables? [WBSCTE 2009]

Answer:
Enumeration Variables: Enumerated data type provides a way for attaching names to numbers. The variables defined for the enumerated data type are called enumerated variables.

19. What are the user defined data types in C? **[WBSCTE 2010]**
Answer: Structure, Enum, Array.

20. What are the self-referential structures? **[WBSCTE 2010]**
Answer:
A self-referential structure is one of the data structures which refer to the pointer to point to another structure of the same type. For example, a linked list is supposed to be a self-referential data structure. The next node of a node is being pointed, which is of the same struct type.

21. How is an array represented in memory? **[WBSCTE 2012, 2013]**
Answer:
An array is a block of contiguous objects with no spaces in between.

22. Distinguish between structure and array. **[WBSCTE 2016]**
Answer:
Structures stores elements like array. The only difference is array stores similar type of elements, but structure can store different types of elements.

Long Answer Type Questions

➲ 1. What is an enumeration? How is enumeration defined? Give example. **[WBSCTE 2005, 2010]**

Answer:
The enumerated data types are user defined data types but they are somewhat different, from the typedef data types. Enumerated data type provides a way for attaching names to numbers, thereby increasing comprehensibility of the code. These data types work when you know in advance a finite list of values that a data type can take on. For example, if you want to define a data type for the days of a week, you know in advance the days of week can have seven values, i.e., Sunday, Monday.....Saturday. Now, we can define an enumerated data type for the above given days.
The variable defined for the enumerated data type can access only the member of that data type; it can't access or assign any other value. The general format for the enumerated data type is:
enum user_define_name { element_1, element_2,.....element_n};
In the above syntax, enum is a keyword telling the compiler about the declaration of enumerated data type. user_define_name is the name given to the enumerated data type.
For the week, we can define the enumerated data type as follows:
enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday} ;
week obj1.

2. Write the basic differences between structure and union in C-language. Also write one use of each of these keywords. [WBSCTE 2006, 2010, 2017]

OR,

Compare structure and union in C mentioning at least three points. [WBSCTE 2015, 2016]

Answer:

Structure	Union
(i) A structure is a collection of variables, referenced under one name, providing a convenient means of keeping related information together.	(i) Closely associated with the structure is the union which also contains multiple members.
(ii) Thus, a single structure might contain integer elements, floating point elements and character elements. Pointers, array and other structure can also be included within the structure.	(ii) Thus, union permits several different data items to be stored in the same portion.
(iii) It is used in linked list data structure.	(iii) It is used in advanced C programming.

3. (a) Differentiate between structure and union. [WBSCTE 2006, 2007, 2017]

(b) Define enumerated data type and explain its use. Give example for this. [WBSCTE 2006, 2007]

Answer:

a) Difference between structure and union:

Structure	Union
(i) A structure is a collection of variables, referenced under one name, providing a convenient means of keeping related information together.	(i) Closely associated with the structure is the union which also contains multiple members.
(ii) Thus, a single structure might contain integer elements, floating point elements and character elements. Pointers, array and other structure can also be included within the structure.	(ii) Thus, union permits several different data items to be stored in the same portion.

b) Enumerated Data Type: The enumerated data types are user defined data types but they are somewhat different, from the typedef data types. Enumerated data type provides a way for attaching names to numbers, thereby increasing comprehensibility of the code. These data types work when you know in advance a finite list of values that a data type can take on. For example, if you want to define a data type for the days of a week, you know in advance the days of week can have seven values, i.e., Sunday, Monday.....Saturday. Now, we can define an enumerated data type for the above given days.

The variable defined for the enumerated data type can access only the member of that data type; it can't access or assign any other value. The general format for the enumerated data type is:

enum user_define_name { element_1, element_2,.....element_n};

In the above syntax, enum is a keyword telling the compiler about the declaration of enumerated data type. user_define_name is the name given to the enumerated data type. For the week, we can define the enumerated data type as follows:

enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday} ;
week obj1;

4. Distinguish between struct and enum data structures. Give example.

[WBSCTE 2008]

Answer:

struct: A structure is a data type that aggregates different types of data. In C, a structure is represented by "struct" keyword. It is used to declare a new data-type. Basically, this means grouping different variables together.

As a simple example, if we want to define a data structure for representing date, then it can be represented by a structure declaration as shown below --

```
struct date
{
        int day;
        int month;
        int year;
};
```

The keyword 'struct' declares a structure to hold the details of three fields, viz., day, month, and year. These are members of structure. Each member may belong to different or same data type.

enum: The enumerated data types are user defined data types, which provides a way for attaching names to numbers, thereby increasing comprehensibility of the code. These data types work when you know in advance a finite list of values that a data type can take on. For example, if you want to define a data type for the days of a week, you know in advance the days of week can have seven values, i.e., Sunday, Monday.....Saturday. Now, we can define an enumerated data type for the above given days.

The variable defined for the enumerated data type can access only the member of that data type; it can't access or assign any other value. 'enum' is a keyword telling the compiler about the declaration of enumerated data type. For the week, we can define the enumerated data type as follows:

enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday} ;

5. Where are structures useful?

[WBSCTE 2008]

Answer:

Arrays are used to store large set of data and manipulate them, but the disadvantage is that all the elements stored in an array are to be of the same data type. If we need to use a collection of different data items, it is not possible using an array. When we require using a collection of different data items of different data types we can use a structure. Structure is a method of encapsulating data of different types. A structure is a convenient method of handling a group of related data items of different data types.

6. How is a structure member accessed? How can a structure member be processed?

[WBSCTE 2008]

Answer:
A structure is a data type that aggregates different types of data. In C, a structure is represented by "struct" keyword. It is used to declare a new data-type. Basically, this means grouping different variables together.
As a simple example, if we want to define a data structure for representing data type books, then it can be represented by a structure declaration as shown below –

```
struct books
{
        char title[20];
        char author[15];
        int pages;
        float price;
};
```

The keyword 'struct' declares a structure to hold the details of four fields, viz., title, author, pages and price. These are members of structure. Each member may belong to different or same data type. Before we can use the type books we have to create a variable of the type books. We can declare structure variables as:

```
struct books book1, book2;
```

The members of a structure should be linked to structure variables in order to make them meaningful members. The link between a member and a variable is established using the member operator '.' This is known as dot operator or period operator. For example, book1.price is the variable representing the price of book1 and can be treated like any other ordinary variable.

➲ 7. Write two features of structures. How can structure variable be declared? **[WBSCTE 2008]**

Answer:
Features of structures:
(i) Structure is a collection of variables, referenced under one name, providing a convenient means of keeping related information together.
(ii) A structure allocates memory for all of its members. Hence, all the members of a structure can be accessed at the same time.

Declaration of structure variable:
We can declare structure variables using the tag name anywhere in the program. For example, for the structure:

```
struct books
{
        char title[20];
        char author[15];
        int pages;
        float price;
};
```

We can declare variables as follows:

```
struct books book1, book2;
```

The above statement declares book1, book2 as variables of type struct books.

8. What is a structure member? What is the relationship between a structure member and a structure? **[WBSCTE 2009]**

Answer:

```
typedef struct {
        char name[64];
        char course[128];
        int age;
        int year;
} student;
```

This defines a new type student variables of type student can be declared as follows.
student st_rec;
Notice how similar this is to declaring an int or float.
The variable name is st_rec, it has members called name, course, age and year.
Each member of a structure can be used just like a normal variable, but its name will be a bit longer. To return to the examples above, member name of structure st_rec will behave just like a normal array of char, however we refer to it by the name

```
st_rec.name
```

Here the dot is an operator which selects a member from a structure.
Where we have a pointer to a structure we could dereference the pointer and then use dot as a member selector. This method is a little clumsy to type. Since selecting a member from a structure pointer happens frequently, it has its own operator -> which acts as follows. Assume that st_ptr is a pointer to a structure of type student We would refer to the name member as

```
st_ptr -> name
```

9. For what kind of applications are unions useful? What is tag? **[WBSCTE 2009]**

Answer:
Unions like structure contain members whose individual data types may differ from one another. However the members that compose a union all share the same storage area within the computer's memory where as each member within a structure is assigned its unique storage area. Thus unions are used to conserve memory. They are useful for application involving multiple members, where values need not to be assigned to all the members at a time. So to conserve memory we use unions.
A "structure declaration" names a type and specifies a sequence of variable values (called "members" or "fields" of the structure) that can have different types. An optional identifier, called a "tag," gives the name of the structure type and can be used in subsequent references to the structure type. We can declare a tag as follows:
typedef struct_data_type tag_name;

10. What are enumerated variables? How are they declared? What is the advantage of using them in a program? **[WBSCTE 2009]**

Answer:
Enumerated variables:
Advantage of enumerated variables: It can help in making the program listings more readable, which can be an advantage when a program gets complicated or when more

than one programmer would be working on it. Using enumerated variables can also reduce programming errors.

Advantage of enumerated variables: It can help in making the program listings more readable, which can be an advantage when a program gets complicated or when more than one programmer would be working on it. Using enumerated variables can also reduce programming errors.

11. Explain the functionality of enumerated data type. [WBSCTE 2010]

Answer:

Enumeration type allows programmer to define their own data type . Keyword enum is used to defined enumerated data type.

enum type_name{ value1, value2,...,valueN };

Here, *type_name* is the name of enumerated data type or tag. And *value1*, *value2*,....,*valueN* are values of type *type_name*.

By default, *value1* will be equal to 0, *value2* will be 1 and so on but, the programmer can change the default value as below:

```
enum suit{
    club=0;
    diamonds=10;
    hearts=20;
    spades=3;
};
```

12. Write a program to display the different parameters of 10 men in a table. The different parameters to be entered are height, weight, age and chest in cms. Find the number of men who satisfy the following conditions. [WBSCTE 2010]

(i) Age should be greater than 18 and less than 25.

(ii) Height should be between 5.2 and 5.6 inches.

(iii) Weight should be in between 45 and 60 kg.

(iv) Chest should be greater than 45 cms.

Answer:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
clrscr( );
struct men
{
float height;
float weight;
int age;
float chest;
}a[10];
int i, count=0;
printf("Enter details of 10 men:-\n");
/*Input for 10 men*/
for(i=0; i<10; i++)
```

```
{
printf("Enter details for man%d--", i+1);
printf("\nHeight in inchs-");
scanf("%f", &a[i].height);
printf("\nWeight in kgs-");
scanf("%f", &a[i].weight);
printf("\nAge-");
scanf("%d", &a[i].age);
printf("\nChest in cms-");
scanf("%f", &a[i].chest);
}
for(i=0; i<10; i++)
{
if((a[i].age>18 && a[i].age<25)                //check for condition
(i)
&& (a[i].height>=5.2 && a[i].height<=5.6) //check for condition
(ii)
&& (a[i].weight>=40 && a[i].weight<=60)    //check for condition
(iii)
&& (a[i].chest>45))                        //check for condition (iv)
{
count++;
}
}
printf("%d men satisfy all the conditions.", count);
getch( );
}
```

13. Write down a program in C to insert 'n' students data in a structure array named as 'Student' consisting of roll, name and marks of three subjects as member data. Display all those data and average marks of each student in tabular format. **[WBSCTE 2016]**

Answer:

```
#include <stdio.h>
struct student
{
    char name[50];
    int roll;
    int sub1;
    int sub2;
    int sub3;
} s[10];
int main()
{
    int i, total;
    float average;
    printf("Enter information of students:\n");
    // storing information
    for(i=0; i<10; ++i)
```

```
    {
        s[i].roll = i+1;
        printf("\nFor roll number%d,\n",s[i].roll);
        printf("Enter name: ");
        scanf("%s",s[i].name);
        printf("\nEnter Marks in Three Subjects = ");
        scanf("%d%d%d",&s[i].sub1,&s[i].sub2,&s[i].sub3);
        printf("\n");
    }
    printf("Displaying Information:\n\n");
    // displaying information
    for(i=0; i<10; ++i)
    {
        printf("\nRoll number: %d\n",i+1);
        printf("Name: ");
        puts(s[i].name);
        total=s[i].sub1+s[i].sub2+s[i].sub3;
        average = total/3;
        printf(" Average Marks: %.1f",average);
        printf("\n");
    }
    return 0;
}
```

14. Write a program using structure to read and display the roll, name and marks of 3 subjects of a student. **[WBSCTE 2017]**

Answer:

```
#include <stdio.h>
#include <conio.h>
typedef struct Student
{
    int roll;
    char name [10];
    int marks [3];
}S;
void main ( )
{
    int n, i;
    S stud;
    //Get the student details.
    printf("\n Enter roll no. of the student:");
    scanf("%d", &stud.roll);
    printf("\n Enter the name of the student:");
    gets(stud.name);
    //Enter 3 marks of the student
    printf("\n Enter marks of 3 subjects");
        for (i=0; i<3; i++)
        {
            printf("Marks %d:", i+1);
```

```
            scanf("%d", & stud.marks[i])
        }
    //print the student details.
    printf("\n Details of student is :\n);
    printf("Roll no: %d\n", stud. Roll);
    printf("Name: %s\n" stud.name);
    for (i=0; i<3; i++)
    {
        printf("Marks %d : %d \n", i+1, stud.marks[i]);
    }
}//End of main.
```

15. Write short notes on the following:

(a) Union **[WBSCTE 2009]**

(b) Enumerated data types. **[WBSCTE 2010, 2011, 2015]**

Answer:

a) Union: Unions like structure contain members whose individua1 data types may differ from one another. However the members that compose a union all share the same storage area within the computer's memory where as each member within a structure is assigned its unique storage area. Thus unions are used to conserve memory. They are useful for application involving multiple members, where values need not to be assigned to all the members at a time. Like structures union can e declared using the keyword union as follows:

```
union item
{
int m;
float p;
char C;
} code;
```

This declares a variable code of type Union. The union contains three members each with a different data type. However we can use only one of them at a time. This is because if only one location is allocated for union variable irrespective of size. The compiler allocates a piece of storage that is large enough to access a union member we can use the same syntax that we use to access structure members. That is—

code.m
code.p
code.c

are all valid member variables.

b) Enumerated data types: Variable of type enum can be created as:

```
enum boolean{
false;
true;
};
enum boolean check;
```

Here, a variable check is declared which is of type enum boolean.

```
#include <stdio.h>
```

```
enum week{ sunday, monday, tuesday, wednesday, thursday, friday,
saturday};
int main(){
enum week today;
today=wednesday;
printf("%d day",today+1);
return 0;
}
```

Enumerated data type variables can only assume values which have been previously declared. Variables of enum type can be used in indexing expressions and as operands of all arithmetic and relational operators. Enumerations provide an alternative to the #define preprocessor directive with the advantages that the values can be generated for you and obey normal scoping rules.

Example:

```
enum DAY              /* Defines an enumeration type     */
{
saturday,             /* Names day and declares a        */
sunday = 0,         /* variable named workday with     */
monday,             /* that type                       */
tuesday,
wednesday,          /* wednesday is associated with 3 */
thursday,
friday
} workday;
```

Unit 7: C Preprocessor

Unit at a glance:

7.1 Introduction

It is a program that processes our source program before it is passed to the compiler. If the source code is stored in a file test.C then the expanded source code gets stored in a file test.I. When this expanded source code is compiled the object code gets stored in test.OBJ. When this object code is linked with the object code of library functions the resultant executable code gets stored in test.EXE.

7.2 Types of preprocessor directives

The preprocessor offers several features called preprocessor directives. Each of these preprocessor directives begin with a # symbol. The directives can be placed anywhere in a program but are most often placed at the beginning of a program, before the first function definition. We would learn the following preprocessor directives here:

(a) Macro expansion
(b) File inclusion
(c) Conditional Compilation
(d) Miscellaneous directives

```
#define FOUND printf ( "hello user" ) ;
main( )
{
char signature ;
if ( signature == 'Y' )
FOUND
else
printf ( "user A welcome" ) ;
}
```

7.3 Rules for using Macros

1. Place no blank spaces in macro definition.#AREA(x) (x*y)
2. Entire macro expansion must be inside parenthesis.
3. Macros can be split into multiple lines, with a '\' (back slash) present at the end of each line.

```
#define LINE for ( i = 0 ; i < 79 ; i++ ) \
printf ( "+", ) ;
```

7.4 Distinction between Macro and Function

1. Size of the program increases for macros whereas for programs the space remains the same.
2. Macros run faster, whereas the program run slower because of return.
3. Macro provides short hand way of replacement of repeating codes, but functions replace the large codes into smaller program modules.
4. Macro is an unintelligent way of module formation than function.

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. '#define' preprocessor statement can be used for defining – [WBSCTE 2007]
   (a) macro (b) function (c) symbolic constants only (d) both (a) % (c)
   Answer: (a)

2. # define max(a, b) a< b? a:b++; [WBSCTE 2010]
   int a ; b ;
   a = 8, b = 9;
   printf("%d", max(a.b));
   The output is –
   (a) 8 (b) 9 (c) 10 (d) compilation error
   Answer: (d)

3. Input/output function prototypes and macros are defined in which header file? [WBSCTE 2012, 2013]
   (a) conio.h (b) stdlib.h (c) stdio.h (d) dos.h
   Answer: (c)

4. The keyword used to transfer control from a function back to the calling function is – [WBSCTE 2012, 2013]
   (a) switch (b) goto (c) go back (d) return
   Answer: (d)

5. How will you print \n on the screen? [WBSCTE 2012, 2013]
   (a) printf("\n") (b) echo\\n (c) printf('\n') (d) printf(\\n)
   Answer: (a)

6. When we mention the prototype of a function? [WBSCTE 2012, 2013]
   (a) Defining (b) Declaring (c) Prototyping (d) Calling
   Answer: (b)

7. How would you round off a value from 1.66 to 2.0? [WBSCTE 2012, 2013]
   (a) ceil(1.66) (b) floor(1.66) (c) roundup(1.66) (d) roundto(1.66)
   Answer: (a)

8. Is the following statement a declaration or definition? [WBSCTE 2012]
   extern int i;
   (a) declaration (b) definition (c) function (d) error
   Answer: (a)

9. Is the following statement a declaration or definition? [WBSCTE 2013]
   extern int i
   (a) Declaration (b) Definition (c) Function (d) Error
   Answer: (a)

B. State whether the following statements are True or False:

10. A file must be explicitly opened before it can be used.

Answer: True [WBSCTE 2013]

11. C is a low level language.

Answer: False [WBSCTE 2013]

12. What will be the output of the following codes? [WBSCTE 2017]

```
#define SQUARE(n) n*n
main()
{
      int j;
      j = 64 / SQUARE (4);
      printf ("j=%d",j);
}
```

Answer: Output : j=4

C. Answer the following questions:

13. What is preprocessor? [WBSCTE 2008, 2015]

Answer:

The C preprocessor is a *macro processor* that is used automatically by the C compiler to transform your program before actual compilation. It is called a macro processor because it allows you to define *macros*, which are brief abbreviations for longer constructs.

14. Name two unconditional preprocessor directives. [WBSCTE 2008]

Answer:

Two unconditional preprocessor directives - #define and #include.

15. What do you mean by pre-processor directive? [WBSCTE 2016]

Answer: ***Refer to article no. 7.2.***

Long Answer Type Questions

1. (a) What is a macro? Mention the similarities and difference between macros and functions. How is a multiline macro defined? Describe the use of arguments within a macro. [WBSCTE 2007, 2008, 2009, 2011]

OR,

What do you mean by macros in C?What are the differences between macros and functions in C? [WBSCTE 2015]

(b) Explain the following preprocessor directives – [WBSCTE 2004, 2005, 2006]

(i) #if [WBSCTE 2004, 2005, 2006, 2007]

(ii) #ifdef [WBSCTE 2004, 2005, 2006]

(iii) #undef [WBSCTE 2007]

(iv) #define [WBSCTE 2007]

(v) #include

Answer:

a) Macros are the identifiers that represent statements or expressions. A *macro* is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro. There are two kinds of macros. They differ mostly in what they look like when they are used. *Object-like* macros resemble data objects when used, *function-like* macros resemble function calls.

Macro: Macro is a one type of C preprocessor directive. By macro we replace a predefined string composed of one or more tokens. Macro also can be used as abbreviations for lengthy and frequently used statements. The one restriction on macros is that, even though their definition can occupy more than one line they cannot include a new line.

Function: A function is a self-contained body that consists of several lines of execution to achieve a certain behavior of a program. The idea behind a function is that it compartmentalizes part of the program and in particular, that code within the function has some useful properties.

Advantages and disadvantages of macro over function:

Advantage: (i) In a macro call the preprocessor replaces the macro template with its macro expansion.

(ii) Macros make the program runs faster.

(iii) Macro is simple.

(iv) There is no type of macro. So type declaration is not mandatory.

Disadvantage: (i) Macro increases program size.

A macro can be defined in terms of other macros. This is called multi-line macro definition.

```
#define CONTROL "%d \n"
#define Print int (x) printf(CONTROL, x)
#define TEST(x) if (x>0) Print int (x)
```

If the program contains the statement TEST(w); where w is an integer variable, the statement goes through the following conversion steps:

(i) if (w>0) Print int(w)

(ii) if (w>0) printf(CONTROL, w)

(iii) if (w>0) printf("%d \n", w)

Function-like macros can take *arguments*, just like true functions. To define a macro that uses arguments, we insert *parameters* between the pair of parentheses in the macro definition that make the macro function-like. The parameters must be valid C identifiers, separated by commas and optionally whitespace.

To invoke a macro that takes arguments, we write the name of the macro followed by a list of *actual arguments* in parentheses, separated by commas. The invocation of the macro need not be restricted to a single logical line—it can cross as many lines in the source file. The number of arguments we give must match the number of parameters in the macro definition. When the macro is expanded, each use of a parameter in its body is replaced by the tokens of the corresponding argument. E.g. –

```
#define sqr(x) (x*x)
```

b) (i) #if: The '#if' directive allows to test the value of an arithmetic expression, rather than the mere existence of one-macro. Its syntax is

#if *expression*

controlled text

#endif /* *expression* */

expression is a C expression of integer type, subject to stringent restrictions.

(ii) #ifdef: This macro is used to represent the condition – if this macro is defined. This is a conditional macro. The simplest sort of conditional is

#ifdef *MACRO*

controlled text

#endif /* *MACRO* */

This block is called a *conditional group*. *controlled text* will be included in the output of the preprocessor if and only if *MACRO* is defined. We say that the conditional *succeeds* if *MACRO* is defined, *fails* if it is not.

(iii) #undef: This command undefined a macro. A macro must be undefined before being redefined to a different value. #undef takes a single argument, the name of the macro to undefine. You use the bare macro name, even if the macro is function-like. It is an error if anything appears on the line after the macro name. '#undef' has no effect if the name is not a macro.

```
#define FOO 4
x = FOO;        ==> x = 4;
#undef FOO
x = FOO;        ==> x = FOO;
```

(iv) #define: A *preprocessor define directive* directs the preprocessor to replace all subsequent occurrences of a macro with specified replacement tokens. The #define directive can contain:

- Object -like macros

An *object-like macro definition* replaces a single identifier with the specified replacement tokens. The following object-like definition causes the preprocessor to replace all subsequent instances of the identifier COUNT with the constant 1000:

```
#define COUNT 1000
```

- Function-like macros

More complex than object-like macros, a function-like macro definition declares the names of formal parameters within parentheses, separated by commas.

```
#define SUM(a,b,c)  a + b + c
```

(v) #include: The include directive instructs the preprocessor to paste the text of the given file into the current file. Generally, it is necessary to tell the preprocessor where to look for header files if they are not placed in the current directory or a standard system directory. If an included file cannot be found, compilation will cease with an error. Its syntax is –

```
#include <header name>
```

2. (a) State the difference between include<stdio.h> and #include "stdio.h". **[WBSCTE 2008, 2009]**

(b) Write a macro named MAX that uses the conditional operator to determine the largest among three integer numbers. **[WBSCTE 2008, 2010]**

Answer:

a) <stdio.h> is a header file which available in include directory of the system. When we write #include<stdio.h> preprocessor search for it in include directory directly and not out of this directory. But when we write "stdio.h" precessor start searching for this header file from current directory and then in parent directories. So if we write our own stdio.h in the current directory and include in program as #include"stdio.h" then our header will be included instead of system header.

b) #define MAX(a,b,c) (a>b)?((a>c)?a:c):((b>c)?b:c)

3. Difference between macro and function. What do you mean by macro expansion? Give example. **[WBSCTE 2009]**

Answer:

Macro: Macro is a one type of C preprocessor directive. By macro we replace a predefined string composed of one or more tokens. Macro also can be used as abbreviations for lengthy and frequently used statements. The one restriction on macros is that, even though their definition can occupy more than one line they cannot include a new line.

Function: A function is a self-contained body that consists of several lines of execution to achieve a certain behavior of a program. The idea behind a function is that it compartmentalizes part of the program and in particular, that code within the function has some useful properties.

Advantages and disadvantages of macro over function:

Advantage:

(i) In a macro call the preprocessor replaces the macro template with its macro expansion.
(ii) Macros make the program runs faster.
(iii) Macro is simple.
(iv) There is no type of macro. So type declaration is not mandatory.

Disadvantage:

(i) Macro increases program size.

Macro expansion: A macro can be defined in terms of another macros. This is also called multi-line macro definition.

```
#define CONTROL "%d \n"
#define Print int (x) printf(CONTROL, x)
#define TEST(x) if (x>0) Print int (x)
```

If the program contains the statement TEST(w); where w is an integer variable, the statement goes through the following conversion steps:

(i) if (w>0) Print int(w)
(ii) if (w>0) printf(CONTROL, w)
(iii) if (w>0) printf("%d \n", w)

4. Predict the output or error(s) for the following and explain. **[WBSCTE 2011]**

```
#define prod(a,b) a*b
main()
{
int x=3, y=4;
printf("%d",prod(x+2,y-1));
}
```

Answer:

Output: 10

Explanation:

This program deals with macros, their side effects and operator precedence. Here prod is a macro which multiplies its two arguments a and b. Let us take a closer look.

prod(a, b) = a*b

prod(x+2, y-1) = x+2*y-1 = 3+2*4-1 = 3+8-1=10

If the programmer really wanted to multiply x+2 and y-1, he should have put parenthesis around a and b as follows.

prod(a,b) = (a)*(b)

This type of mistake in macro definition is called – macro side-effects.

5. (a) Explain the importance of the # define preprocessor directive with example.

(b) Distinguish between macro and function with example. **[WBSCTE 2012]**

Answer:

a) A *preprocessor define directive* directs the preprocessor to replace all subsequent occurrences of a macro with specified replacement tokens. The #define directive can contain:

- **Object -like macros:** An *object-like macro definition* replaces a single identifier with the specified replacement tokens. The following object-like definition causes the preprocessor to replace all subsequent instances of the identifier COUNT with the constant 1000:

  #define COUNT 1000

- **Function-like macros:** More complex than object-like macros, a function-like macro definition declares the names of formal parameters within parentheses, separated by commas.

  #define SUM(a,b,c) a + b + c

b) ***Macro:*** Macro is a one type of C preprocessor directive. By macro we replace a predefined string composed of one or more tokens. Macro also can be used as abbreviations for lengthy and frequently used statements. The one restriction on macros is that, even though their definition can occupy more than one line they cannot include a new line.

Function: A function is a self-contained body that consists of several lines of execution to achieve a certain behavior of a program. The idea behind a function is that it compartmentalizes part of the program and in particular, that code within the function has some useful properties.

6. Define 'inline function'. **[WBSCTE 2015]**

Answer:

An inline function is one for which the compiler copies the code from the function definition directly into the code of the calling function rather than creating a separate set of instructions in memory. Instead of transferring control to and from the function code segment, a modified copy of the function body may be substituted directly for the function call. In this way, the performance overhead of a function call is avoided.

A function is declared inline by using the inline function specifier or by defining a member function within a class or structure definition. The inline specifier is only a suggestion to the compiler that an inline expansion can be performed; the compiler is free to ignore the suggestion.

The following code fragment shows an inline function definition.

inline int add(int i, int j) { return i + j; }

7. Write a short note on Macro. **[WBSCTE 2011]**

Answer:

A *macro* is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro. There are two kinds of macros. They differ mostly in what they look like when they are used. *Object-like* macros resemble data objects when used, *function-like* macros resemble function calls. Any valid identifier can be defined as a macro, even if it is a C keyword.

As an example, here is a macro that computes the minimum of two numeric values, as it is defined in many C programs, and some uses.

```
#define min(X, Y)  ((X) < (Y) ? (X) : (Y))
  x = min(a, b);          ==> x = ((a) < (b) ? (a) : (b));
  y = min(1, 2);          ==> y = ((1) < (2) ? (1) : (2));
  z = min(a + 28, *p);    ==> z = ((a + 28) < (*p) ? (a + 28) : (*p));
```

Unit 8: File Handling

Unit at a glance:

The file name can be indicated by a file pointer. Compiler always uses this pointer to refer this file instead of using name. The file pointer can be declared in the following manner:

```
FILE _ fptr;
```

where fptr is a special type of pointer variable to store the address of a file.

It points to information that defines various things about the file, including its name, status and the current position of the file.

10.3 File

8.1 Opening a file pointer using fopen

In order to perform any sort of operation in a file, the file should be opened in the beginning of the program. To open a file, the function 'fopen()' is used.

This function returns a pointer to a file. The syntax is given below:

```
file_pointer= fopen ( " file _ name " , "mode " ) ;
```

file_name

8.2 Closing a file using fclose

The fclose() method can be used to close a file pointer from a file. This is usually done so that the pointer can be used to access a different file. Systems have a limit on the number of files, which can be open simultaneously, so it is a good idea to close a file when you have finished using it. This would be done using a statement like

```
fclose(file_pointer)
fclose(fptr);
```

Short Answer Type Questions

A. Choose the correct answer from the given alternatives in each of the following:

1. A file is declared as a pointer of data type – **[WBSCTE 2007]**
   (a) int (b) float (c) string (d) string (e) none of these

Answer: (d) (file is declared as a pointer of file stream).

2. A file opend in "w+" mode can be – **[WBSCTE 2007]**
   (a) read/write (b) read only (c) write only (d) none of these

Answer: (a)

3. When fopen( ) fails to open a file it returns **[WBSCTE 2007]**
   (a) NULL (b) – 1 (c) 1 (d) none of these

Answer: (a)

4. the EOF is equivalent to – [WBSCTE 2007]
(a) – 1 (b) 0 (c) 1 (d) none of these
Answer: (a)

5. Command line arguments are used to accept argument from – [WBSCTE 2007]
(a) command prompt of operating system (b) through scanf ( ) statement
(c) both (a) & (b) (d) none of these
Answer: (a)

6. The function ftell( ) – [WBSCTE 2007]
(a) reads a character from a file (b) reads an integer from a file
(c) gives the current position in the file (d) none of the above
Answer: (c)

7. The function rewind( ) – [WBSCTE 2008, 2009]
(a) reads a character from a file (b) writes a character to a file
(c) sets the position to the beginning of the file (d) none of these
Answer: (c)

8. The function ftell ( ) – [WBSCTE 2010]
(a) reads a character from a file (b) reads an integer from a file
(c) gives the current position in the file (d) none of the above
Answer: (c)

B. State whether the following statements are True or False:

9. A binary file has no end of file marker. [WBSCTE 2006, 2008]
Answer: False (Each file has end of file marker.)

10. Binary file has end-of-file marker code, but not in text file. [WBSCTE 2007]
Answer: False (Each file has end of file marker.)

11. "r+b" string is the file opening string and the file can be opened in text mode and the file can be readable and writable. [WBSCTE 2007]
Answer: False (The file is opened in binary mode)

12. '*argv[ ]' and 'argc' are used for passing command line arguments to a C language program. [WBSCTE 2007]
Answer: True (E.g.- main(int argc, char *argv[ ])

13. All files must be explicitly closed. [WBSCTE 2008, 2009]
Answer: True (Otherwise, the file may save unnecessary data.)

14. All files must be explicitly closed – [WBSCTE 2009]
Answer: True

15. "wt" is a valid file opening mode in C. [WBSCTE 2010]
Answer: False

16. "SEEK_SET" is the current position of the file pointer. [WBSCTE 2010]

Answer: True

17. "EOF" is a ASCII code for end of file marker. [WBSCTE 2010]
Answer: True

18. A file must be explicitly opened before it can be used. [WBSCTE 2010]
Answer: True (Otherwise, the file cannot be read or write.)

19. The function fseek will return 0 if we attempt to move file pointer beyond the file boundaries. [WBSCTE 2010]
Answer: True

C. Answer the following questions:

20. How are the command line arguments passed in C? [WBSCTE 2005, 2010]
Answer:
The syntax of command line argument:
int main (int argc, char * argv [ ])
where, argc is the no. of arguments passed to main, and agrv[ ] are the arguments passed to the main.

21. What is the purposes of fopen ( ) function? [WBSCTE 2008]
Answer: fopen( ) opens a file in different modes.

22. What is EOF? [WBSCTE 2008, 2009]
Answer: EOF is End of file. It is a marker, used to denote the end of file.

23. How does an append mode differ from a write mode? [WBSCTE 2009, 2017]
Answer:
In append mode, the previous data exists, and the new data is inserted the end of previous data. In write mode, all previous data are overwritten by the new data.

24. Distinguish between printf( ) and fprintf( ). [WBSCTE 2009, 2017]
Answer:
printf( ) displays characters on screen; whereas fprint( ) writes characters to file.

25. What do you mean by the statement fp=fopen("Sample", r+); where fp is a file pointer. [WBSCTE 2016]
Answer:
To open a file, the function 'fopen( )' is used. This function returns to afile. The syntax is given below:-
file_pointer = fopen("file_name", "mode");
So here file 'Sample' is opened in read mode.

Long Answer Type Questions

1. What do you mean by command line argument? Write a command line argument based C-program which will check whether required number of arguments given or not and the program will perform all basic calculator operation s (+, -, *, /) given operands and operates as command line arguments forms. [WBSCTE 2007, 2009, 2010]

Answer:

It is possible to pass arguments to C programs when they are executed. The brackets which follow main are used for this purpose. *argc* refers to the number of arguments passed, and *argv[]* is a pointer array which points to each argument which is passed to main.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main(int agrc, char *argv[ ])
{
clrscr( );
float a, b, ans;
char c;
int argc =argc;
if(argc!=4)
{
printf("Required no. of arguments are not supplied.");
exit(0);
}
c=*argv[3];
a=atof(argv[1]);
b=atof(argv[2]);
if(c=='+')
ans = a+b;
else if(c=='-')
ans = a-b;
else if(c=='*')
ans = a*b;
else if(c=='/')
if(b!=0)
ans = a/b;
getch( );
}
```

2. Write the use of the following library function –

(i) fopen( ) [WBSCTE 2008]

(ii) ftell( ) [WBSCTE 2009]

(iii) fseek( ) [WBSCTE 2009]

Answer:

(i) **fopen( ):** In order to perform any sort of operation in afile , the file should be opened in the beginning of the program. To open a file, the function 'fopen( )' is used. This function returns to afile. The syntax is given below:-

file_pointer = fopen("file_name", "mode");

(ii) **ftell( ):** ftell( ) function is used to find out the current position of the file pointer in a file. The syntax of the function is given as:-

position = ftell(file_pointer);

(iii) **fseek( ):** To search for a record we are to move the file pointer to the desired offset position in the first, and then we can read the record for the purpose of processing. To search for a record we use the function fseek( ).

The syntax for the fseek( ) function is:-

fseek(file_pointer, position, SEEK_SET);

➲ 3. Difference between text mode and binary mode of a file. **[WBSCTE 2008, 2009, 2010]**

Answer:

Text mode: The text ("a") mode is used to add records at the end of an existing file.

Binary mode: The binary mode("w") is used to write records in a new file. If existing file is opened in this mode, then the contents of that file will be automatically erased.

➲ 4. Write a program to compare two files specified by the user. **[WBSCTE 2008]**

Answer:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main( )
{
clrscr( );
FILE *f1,*f2;
int flag=0;
char c=EOF, d, n1[16], n2[16];
printf("Enter 1st filename:");
gets(n1);
printf("Enter 2nd filename:");
gets(n2);
f1=fopen(n1,"r");          ////open file_1
f2=fopen(n2,"r");          ////open file_2
if(f1!='\0' && f2!='\0')
{
c=fgetc(f1);
d=fgetc(f2);
flag=-1;
}
```

```
while(c!=EOF)
{
if(c!=d)
{
printf("File contents are not same.");
getch( );
exit(0);
}
c=fgetc(f1);
d=fgetc(f2);
}
fclose(f1);
fclose(f2);
if(flag==-1)
printf("File contents matched.");
getch( );
}
```

➲ 5. What are the common uses of rewind and ftell functions? **[WBSCTE 2009]**

Answer:

Use of rewind( ): Sets the position indicator associated with *stream* to the beginning of the file.

A call to rewind is equivalent to fseek (stream , 0L , SEEK_SET).

Syntax: void rewind (FILE * stream);

Use of ftell( ):Returns the current value of the position indicator of the *stream*. For binary streams, the value returned corresponds to the number of bytes from the beginning of the file.

For text streams, the value is not guaranteed to be the exact number of bytes from the beginning of the file, but the value returned can still be used to restore the position indicator to this position using fseek().

Syntax: long int ftell (FILE * stream);

➲ 6. Discuss different possible modes in which a file can be opened? **[WBSCTE 2009, 2010]**

OR,

What are the different kinds of mode available in C to open a file? **[WBSCTE 2015, 2016]**

Answer:

A file can be opened in various ways for different purposes. This mode decides the read/write operation with the data file.

Mode	Purpose
r	Opens a text file for reading
w	Creates a text file for writing
a	Appends with existing contents in a text file
rb	Opens a binary file for reading
wb	Creates a binary file for reading

ab	Reads a string from a file
r+	Open a file for update both reading and writing
w+	Creates and opens a text file for read/write
a+	Appends or creates a text file for read/append
rb+	Opens a binary file for read/write
wb+	Creates a binary file for read/write
ab+	Appends or creates a binary file for read/write

7. What is the purpose of the fclose function? What is meant by opening a data file? How is this accomplished? **[WBSCTE 2009]**

Answer:

When we have finished working from the file, we need to close it. This done using the function fclose( ).

Syntax:- fclose(file_pointer);

In order to perform any sort of operation in afile , the file should be opened in the beginning of the program. To open a file, the function 'fopen( )' is used. This function returns to afile. The syntax is given below:-

file_pointer = fopen("file_name", "mode");

8. What is the purpose of the library function feof( )? **[WBSCTE 2009]**

Answer:

EOF (End of File) is a special character whose ASCII value is 26, is inserted after the last character in the file to mark the end of file. If this character is detected at any point in the file, the read function would return the EOF signal to the program. In many cases, feof( ) returns a zero if end of file is not reached. When the end of file is reached feof( ) returns a non-zero value.

The syntax of the function is:-

feof(file_pointer);

9. Write a C program to copy one file to another. **[WBSCTE 2010, 2017]**
Both source and destination file names are given as command line arguments. **[WBSCTE 2010]**

Answer:

```
#include<stdio.h>
#include<conio.h>
main(int argc,char *argv[])    /*command line arguments*/
{
FILE *f1,*f2;
char c;
f1=fopen(argv[1],"r");   /*fie for reading*/
f2=fopen(argv[2],"w");   /*file for writing*/
c=fgetc(f1);                    /*copy of file*/
while(c!=EOF)
{
fputc(c,f2);
```

```
c=fgetc(f1);
}
fclose(f1);
fclose(f2);
getch();
}
```

10. How is the random access to files achieved in C? **[WBSCTE 2010]**

Answer:

Normally files and streams are read and written sequentially. However, it's also possible to jump to a certain position in a file.

To jump to a position, it's generally necessary to have been there once already. First, we use the function ftell( ) to find out what its position in the file is; then, later we can use the function fseek to get back to a saved position.

File position are stored as long int. to recorded a position, we would use code like –

long int pos;

pos = ftell(fp);

Later, we could "seek" back to that position with

fseek(fp, pos, SEEK_SET);

11. Write a C program to find the number of alphabets (a – z and A – Z) in a file. **[WBSCTE 2010]**

Answer:

```
#include<stdio.h>
#include<conio.h>
main( )
{
FILE *f1;
int count=0;
char c, filename[16];
printf("Enter filename:");
gets(filename);
f1=fopen(filename,"r"); /*fie for reading*/
c=fgetc(f1);
while(c!=EOF)
{                                /*count for alphabets*/
if((c>='a' && c<='z') || (c>='A' && c>='Z'))
count++;
c=fgetc(f1);
}
fclose(f1);
printf("Total no. of alphabets in the file %s: %d", filename,
count);
getch( );
}
```

12. Write a program in C to create three different files "NUMBER", "EVEN" and "ODD". Now insert some integer numbers randomly into "NUMBER" file. Now from all those integers put all even numbers into "EVEN" and odd numbers into "ODD" file respectively. **[WBSCTE 2015, 2016]**

Answer:

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *f1,*f2,*f3;
int number,i;
clrscr();
printf("Contents of DATA file\n\n");
f1 = fopen("DATA","w");   /* create a data file */
for(i=1;i<=30;i++)
{
scanf("%d",&number);
if(number==-1)break;
putw(number,f1);
}
fclose(f1);
f1 = fopen("DATA","r");
f2 = fopen("ODD","w");
f3 = fopen("EVEN","w");
while((number = getw(f1)) != EOF) /* Read from Data file */
{
if(number%2==0)
putw(number,f3);
else
putw(number,f2);
}
fclose(f1);
fclose(f2);
fclose(f3);
f2 = fopen("ODD","r");
f3 = fopen("EVEN","r");
printf("\n\nContents of ODD file \n\n");
while((number = getw(f2)) != EOF)
printf("%4d",number);
printf("\n\nContents of EVEN file \n\n");
while((number = getw(f3)) != EOF)
printf("%4d",number);
fclose(f2);
fclose(f3);
getch();
}
```

Question Paper 2015 (December)

1. A. Attempt the below questions:

i) What do you mean by size of () operator?

See Unit: Basics of C, Question No. 153 of Short Answer Type Questions.

ii) What is header file?

See Unit: Basics of C, Question No. 154 of Short Answer Type Questions.

iii) Define compiler directive statement.

See Unit: Basics of C, Question No. 155 of Short Answer Type Questions.

iv) What do you mean by operator precedence?

See Unit: Basics of C, Question No. 156 of Short Answer Type Questions.

v) Define typecasting.

See Unit: Basics of C, Question No. 157 of Short Answer Type Questions.

vi) What is the difference between 'a' and "a"?

See Unit: Basics of C, Question No. 158 of Short Answer Type Questions.

vii) Describe prototype function declaration?

See Unit: Functions, Question No. 22 of Short Answer Type Questions.

viii) What do you mean by recursion?

See Unit: Functions, Question No. 23 of Short Answer Type Questions.

ix) Define null pointer.

See Unit: Pointers, Question No. 11 of Short Answer Type Questions.

x) What do you mean by pre-processor?

See Unit: C-Preprocessor, Question No. 12 of Short Answer Type Questions.

B. Choose the correct alternative to complete the following sentences:

a) The library function strcmp("abc","ABC"); returns

(a) 32

(b) –32

(c) 0

✓ (d) 1

Question Papers

b) The variables declared in a structure definition are called its

(a) group variables
(b) merged variables
✓ (c) member variables
(d) total variables

c) By default a variable is assigned with _____ in static storage class

(a) garbage
✓ (b) 1
(c) 0
(d) 10

d) The output of the following program segment will be:

```
void main()
{
int x=4,y=3,z;
z=x---y; printf("\n%d%d%d",x,y,z);
}
```

(a) 422
✓ (b) 331
(c) 432
(d) 222

e) Write the output of the following program segment:

```
void main()
{
int i=1,j=1;
for(;;)  {
if(i>5) break; else j+=i; printf("%d",j); i+=j;
}
}
```

(a) 23
(b) 32
(c) 33
(d) compilation error

Answer: Output will be '2 5'

f) Write down the output of the following program segment(s):

```
void main() {
int i;
for(i=0;i<=10;i++); printf("%d",i);
}
```

(a) display 1 to 10
(b) 10
✓ (c) 11
(d) compilation error

2. a) What do you mean by parameter passing in C?
b) Discuss different types of parameter passing supported by C through appropriate examples.
See Unit: Functions, Question No. 18 of Long Answer Type Questions.

3. a) Write a program in C to find the sum of the series $1^2+(1^2+2^2)+(1^2+2^2+3^2)+\ldots$ up to n terms.
b) Distinguish between entry controlled and exit controlled loop structure mentioning at least three points.
See Unit: Decision Control, Question No. 29 of Long Answer Type Questions.

4. a) Distinguish between 'break' and 'continue' statements.
b) Why 'goto' statement should be avoided in a C program?
c) Write a program in C to find all prime numbers within a given range.
a) See Unit: Decision Control, Question No. 2 of Long Answer Type Questions.
b) See Unit: Decision Control, Question No. 1 of Long Answer Type Questions.
c) See Unit: Decision Control, Question No. 30 of Long Answer Type Questions.

5. a) Write C programs to implement i) strcmp(), ii) strcat() library function by yourself.
b) What do you mean by dynamic memory allocation? How this can be implemented in C.
a) See Unit: Arrays and Strings, Question No. 14 of Long Answer Type Questions.
b) See Unit: Pointers, Question No. 9 of Long Answer Type Questions.

6. a) Define storage classes in C. Give a comparative study between each of them.
b) Write short note on enumerated data type in C.
a) See Unit: Basics of C, Question No. 6 of Long Answer Type Questions.
b) See Unit: Structures & Unions and Enumerated Data Type, Question No. 15.b) of Long Answer Type Questions.

7. a) Compare structure and union in C mentioning at least three points.
b) Define a structure to store the name, roll, an array marks[] which stores marks of three different subjects and a character grade on total marks. Write a program to display the details of the 'n' (given from the keyboard) student in tabular form with grade as per rules – i) 75% and above Grade 'A', ii) 60% to less than 75% Grade 'B', iii) Less than 60% Grade C.
a) See Unit: Structures & Unions and Enumerated Data Type, Question No. 2 of Long Answer Type Questions.
b) See Unit: Arrays and Strings, Question No. 15 of Long Answer Type Questions.

8. a) Create a one dimensional integer array. Now insert n integers randomly into it. Now find the greatest integer from the array using a pointer.
b) What are the drawbacks of pointer?
See Unit: Pointers, Question No. 10 of Long Answer Type Questions.

9. a) What do you mean by macros in C?
b) What are the differences between macros and functions in C?
c) Define 'inline function'.
a) & b) See Unit: C Preprocessor, Question No. 1 of Long Answer Type Questions.

c) See Unit: C Preprocessor, Question No. 7 of Long Answer Type Questions.

10. a) What are the different kinds of mode available in C to open a file?

b) Write a program in C to create three different files "NUMBER", "EVEN" and "ODD". Now insert some integer numbers randomly into "NUMBER" file. Now from all those integers put all even numbers into "EVEN" and odd numbers into "ODD" file respectively.

a) See Unit: File Handling, Question No. 6 of Long Answer Type Questions.

b) See Unit: File Handling, Question No. 12 of Long Answer Type Questions.

Question Paper 2016 (December)

1. A. Attempt the below questions:

i) What do you mean by debugging?

See Unit: Basics of C, Question No. 159 of Short Answer Type Questions.

ii) Define dynamic memory allocation.

See Unit: Pointers, Question No. 9 of Long Answer Type Questions.

iii) What do you mean by type defining in C?

See Unit: Basics of C, Question No. 160 of Short Answer Type Questions.

iv) What do you mean by storage Class?

See Unit: Basics of C, Question No. 161 of Short Answer Type Questions.

v) Define extern storage class.

See Unit: Basics of C, Question No. 162 of Short Answer Type Questions.

vi) What do you mean by ternary operator?

See Unit: Basics of C, Question No. 118 of Short Answer Type Questions.

vii) Distinguish between gets(name) and scanf("%s", name) where 'name' is a character array.

See Unit: Arrays and Strings, Question No. 42 of Short Answer Type Questions.

viii) Write down four binary operators of C.

See Unit: Basics of C, Question No. 163 of Short Answer Type Questions.

ix) Define null pointer.

See Unit: Pointers, Question No. 11 of Short Answer Type Questions.

x) Distinguish between getche() and getch() function.

See Unit: Basics of C, Question No. 164 of Short Answer Type Questions.

xi) Distinguish between structure and array.

See Unit: Structures & Unions and Enumerated Data Type, Question No. 22 of Short Answer Type Questions.

xii) What do you mean by the statement fp=fopen("Sample", r+); where fp is a file pointer.

See Unit: File Handling, Question No. 25 of Short Answer Type Questions.

xiii) What do you mean by pre-processor directive?

See Unit: C Preprocessor, Question No. 14 of Short Answer Type Questions.

xiv) Define L – value of an expression.

See Unit: Basics of C, Question No. 165 of Short Answer Type Questions.

xv) Distinguish between malloc() and calloc() function.

See Unit: Pointers, Question No. 21 of Short Answer Type Questions.

2. a) Write down at least three differences between entry controlled and exit controlled loop structure.

b) Write down a C program to find all the prime numbers in a given range (upper and lower range should be given from the keyboard).

a) See Unit: Decision Control, Question No. 29(b) of Long Answer Type Questions.

b) See Unit: Decision Control, Question No. 30 of Long Answer Type Questions.

3. a) Write a program in C to find the sum of the series 1+(1+2)+(1+2+3)+..... up to 'n' terms, the value of 'n' should be given from the keyboard.

b) Why we should avoid 'goto' statement from our program?

a) See Unit: Decision Control, Question No. 31 of Long Answer Type Questions.

b) See Unit: Decision Control, Question No. 1 of Long Answer Type Questions.

4. a) What do you mean by pointer in C?

b) What are the advantages of pointer variable over memory variable?

c) Write a program in C to access the elements of a one dimensional array using pointer.

a) See Unit: Pointers, Question No. 22 of Short Answer Type Questions.

b) See Unit: Pointers, Question No. 11(a) of Long Answer Type Questions.

c) See Unit: Pointers, Question No. 11(b) of Long Answer Type Questions.

5. a) Write C programs to implement i) strcpy(), ii) strlen() library function by yourself.
b) What do you mean by null character? Describe its uses in string manipulation.
See Unit: Pointers, Question No. 12 of Long Answer Type Questions.

6. a) Distinguish between Structure and union.
b) Write down a program in C to insert 'n' students data in a structure array named as 'Student' consisting of roll, name and marks of three subjects as member data. Display all those data and average marks of each student in tabular format.
a) See Unit: Structures & Unions and Enumerated Data Type, Question No. 2 of Long Answer Type Questions.
b) See Unit: Structures & Unions and Enumerated Data Type, Question No. 13 of Long Answer Type Questions.

7. a) What do you mean by parameter passing in C?
b) Explain different types of parameter passing supported by C using suitable examples.
See Unit: Functions, Question No. 18 of Long Answer Type Questions.

8. a) What are the advantages of switch – case statements over nested if – else structure?
b) Write down a program in C to express any multiword string into its equivalent abbreviated form separated by a dot(.) i.e., SWAPAN KUMAR SEN will be written as S.K.S.
a) See Unit: Decision Control, Question No. 28 of Long Answer Type Questions.
b) See Unit: Arrays and Strings, Question No. 16 of Long Answer Type Questions.

9. a) Create a two dimensional integer array to insert some integers randomly into it. Now find the greatest and smallest integer from the array and also find their corresponding locations.
b) What do you mean by pointer to a function? Give an example.
See Unit: Pointers, Question No. 13 of Long Answer Type Questions.

10. a) What are the different kinds of modes available in C to open a file?
b) Write a program in C to create three different files "NUMBER", 'EVEN" and "ODD". Now insert some integer numbers randomly into "NUMBER" file. Now from all those integers put al even numbers into "EVEN" and odd numbers into "ODD" file respectively.
a) See Unit: File Handling, Question No. 6 of Long Answer Type Questions
b) See Unit: File Handling, Question No. 12 of Long Answer Type Questions.

Question Paper 2017 (December)

1. A. Choose the correct answer from the given alternatives (any ten):

i) C is a
✓(a) Middle level language
(b) High level language
(c) Low level language
(d) none of these

ii) If a is an integer variable, $a = 5/2$, will return a value
(a) 2.5
(b) 3
✓ (c) 2
(d) 0

iii) The break statement is used to exit from
(a) an if statement
✓ (b) a for loop
(c) a program
(d) the main() function

iv) A do while loop is useful when we want that the statements within the loop must be executed
(a) only once
✓ (b) at least once
(c) more than once
(d) none of these above

v) Array is a collection of
(a) different data types scattered throughout memory
(b) the same data type scattered throughout memory
✓ (c) the same data type placed next to each other in memory
(d) different data types placed next to each other in memory

vi) If an array is declared as arr[]={1,2,5,7,9}; then what is the value of size of (arr[3])?
(a) 1
(b) 2
(c) 3
(d) 8

Answer: ***None of these. (7 is the answer)***

vii) ASCII code for a – z ranges from
(a) 0 – 26
(b) 35 – 81
✓ (c) 97 – 123
(d) none of these

viii) s1="H1", s2="HELLO", s3="BYE". How can we concatenate the three strings?
(a) strcat(s1,s2,s3)
✓ (b) strcat(s1,strcat(s2,s3))
(c) strcpy(s1,s2,s3)
(d) strcpy(s1,strcpy(s2,s3))

ix) A string is terminated by a
✓(a) \0
(b) \n
(c) 0
(d) \

x) The default storage class of a local variable is
✓ (a) auto
(b) static
(c) register
(d) extern

xi) *(&num) is equivalent to writing
(a) &num (b) *num ✓ (c) num (d) none of these

xii) A structure can be placed within another structure is known as
✓ (a) nested structure (b) self-referential structure
(c) parallel structure (d) pointer to structure

xiii) From which standard stream does a C program read data?
✓ (a) stdin (b) stdout (c) stderr (d) all of these

B. What will be the output of the following codes? (any five)

i)
```
#define SQUARE(n) n*n
main()
{
    int j;
    j = 64 / SQUARE (4);
    printf ("j=%d",j);
}
```

See Unit: C Preprocessor, Question No. 12 of Short Answer Type Questions.

ii)
```
main()
{
    int arr[]={1,2,3,4,5};
    int *ptr,i;
    ptr=arr+4;
    for(i=0;i<5;i++)
        printf("%d",*(ptr-i));
}
```

See Unit: Pointers, Question No. 24 of Short Answer Type Questions.

iii)
```
intfunc();
main()
{
    printf("%d",func());
    printf("%d",func());
    printf("%d",func());
}
int func()
{
    int counter=5;
    counter++;
    return(counter);
```

```
}
```

See Unit: Functions, Question No. 24(a) of Short Answer Type Questions.

iv)
```
int f2(int a)
{
        return(a*a);
}
int f1(int a, int b)
{
                Return(f2(a*b));
}
main()
{
        int x;
        x=f1(2,4);
        printf("%d",x);
}
```

See Unit: Functions, Question No. 24(b) of Short Answer Type Questions.

v)
```
main()
{
        int i=1;
        while(8)
        {
                printf("%d\n",i++);
                if(i>10)
                        break;
        }
}
```

See Unit: Decision Control, Question No. 35(a) of Short Answer Type Questions.

vi)
```
main()
{
        int x=3,y,z;
        y=x=10;
        z=x<10;
        printf("x=%d y=%d z=%d",x,y,z);
}
```

See Unit: Basics of C, Question No. 167 of Short Answer Type Questions.

vii)
```
main()
{
        int k,num=30;
```

```
    k=(num>5?(num<=10?100:200):500);
    printf("%d",k);
}
```

See Unit: Decision Control, Question No. 35(b) of Short Answer Type Questions.

2. a) What do you mean by associativity property of an operator?
b) Differentiate between if-else if and switch statement.
c) Write a program to check if a character entered through keyboard is a vowel or consonant using switch case statement.

a) See Unit: Basics of C, Question No. 11 of Long Answer Type Questions.

b) & c) See Unit: Decision Control, Question No. 33 of Long Answer Type Questions.

3. a) Two numbers are entered through keyboard. Write a program to find the value of one number raised to the power of another.
b) Differentiate between gets() and scanf().
c) What do you mean by scope and lifetime of a variable?

a) See Unit: Functions, Question No. 12 of Long Answer Type Questions.

b) See Unit: Arrays and Strings, Question No. 42 of Short Answer Type Questions.

c) See Unit: Basics of C, Question No. 166 of Short Answer Type Questions.

4. a) Differentiate between formal parameters and actual parameters.
b) What is prototype declaration of a function?
c) Write a recursive function to get GCD of two numbers.

a) & c) See Unit: Functions, Question No. 19 of Long Answer Type Questions.

b) See Unit: Functions, Question No. 18 of Short Answer Type Questions.

5. a) Write the advantages of using pointers.
b) Explain the term dynamic memory allocation.
c) How are arrays related to pointers?

a) See Unit: Pointers, Question No. 11(a) of Long Answer Type Questions.

b) See Unit: Pointers, Question No. 9 of Long Answer Type Questions.

c) See Unit: Pointers, Question No. 23 of Short Answer Type Questions.

6. a) Differentiate between structure and union.
b) Write a program using structure to read and display the roll, name and marks of 3 subjects of a student.

a) See Unit: Structure & Unions and Enumerated Data Type, Question No. 3(a) of Long Answer Type Questions.

b) See Unit: Structure & Unions and Enumerated Data Type, Question No. 14 of Long Answer Type Questions.

7. a) Distinguish between printf() and fprintf().
b) How does an append mode differ from a write mode in a file?
c) Write a program to copy the content of one file to another.

a) See Unit: File Handling, Question No. 24 of Short Answer Type Questions.
b) See Unit: File Handling, Question No. 23 of Short Answer Type Questions.
c) See Unit: File Handling, Question No. 9 of Long Answer Type Questions.

8. a) What is the difference between while loop and do-while loop? Explain with example.
b) What is goto statement and why this should be avoided?
c) How is a two-dimensional array represented in memory?

a) See Unit: Decision Control, Question No. 32 of Long Answer Type Questions.
b) See Unit: Decision Control, Question No. 1 of Long Answer Type Questions.
c) See Unit: Arrays & Strings, Question No. 35 of Short Answer Type Questions.