

C-PROGRAMMING AND DATA STRUCTURES

UNIT-I

Predict the output or error(s) for the following:

```
1. main()
{
    int i=-1,j=-1,k=0,l=2,m;
    m=i++&& j++&& k++||l++;
    printf("%d %d %d %d %d",i,j,k,l,m);
}
```

Answer:

0 0 1 3 1

```
2. main()
{
    int i=3;
    switch(i)
    {
        default:printf("zero");
        case 1: printf("one");
            break;
        case 2:printf("two");
            break;
        case 3: printf("three");
            break;
    }
}
```

Answer :

three

```
3. main()
{
    printf("%x",-1<<4);
}
```

Answer:

fff0

```
4. main()
{
    int c=- -2;
    printf("c=%d",c);
}
```

Answer:

c=2;

```
5. main()
{
    int i=10;
    i=!i>14;
    Printf ("i=%d",i);
}
```

Answer: i=0

6. main()

```
{  
    printf("\nab");  
    printf("\bsi");  
    printf("\rha");  
}
```

Answer:

hai

7. main()

```
{  
    int i=5;  
    printf("%d%d%d%d%d",i++,i--,++i,--i,i);  
}
```

Answer:

45545

8. main()

```
{  
    printf("%p",main);  
}
```

Answer:

Some address will be printed

9. main()

```
{  
    int i=400,j=300;  
    printf("%d..%d");  
}
```

Answer:

400..300

10. void main()

```
{  
    int i=5;  
    printf("%d",i++ + ++i);  
}
```

Answer:

Output Cannot be predicted exactly.

11. void main()

```
{  
    int i=5;  
    printf("%d",i+++++i);  
}
```

Answer:

Compiler Error

```

12. #include<stdio.h>
    main()
    {
    int i=1,j=2;
    switch(i)
    {
    case 1: printf("GOOD");
            break;
    case j: printf("BAD");
            break;
    }
    }

```

Answer:

Compiler Error: Constant expression required in function main.

```

13. main()
    {
    int i;
    printf("%d",scanf("%d",&i)); // value 10 is given as input here
    }

```

Answer:

1

```

14. main()
    {
    int i=0;

    for(;i++;printf("%d",i) );
        printf("%d",i);
    }

```

Answer:

1

```

15. main()
    {
    printf("%d", out);
    }
    int out=100;

```

Answer:

Compiler error: undefined symbol out in function main.

```

16. main()
    {
    int i=-1;
    +i;
    printf("i = %d, +i = %d \n",i,+i);
    }

```

Answer:

i = -1, +i = -1

17. main()

```
{
char not;
not=!2;
printf("%d",not);
}
```

Answer:

0

18. main()

```
{
int k=1;
printf("%d==1 is ""%s",k,k==1?"TRUE":"FALSE");
}
```

Answer:

1==1 is TRUE

19. main()

```
{
int y;
scanf("%d",&y); // input given is 2000
if( (y%4==0 && y%100 != 0) || y%100 == 0 )
printf("%d is a leap year");
else
printf("%d is not a leap year");
}
```

Answer:

2000 is a leap year

20. main()

```
{
int i=-1;
-i;
printf("i = %d, -i = %d \n",i,-i);
}
```

Answer:

i = -1, -i = 1

21. #include<stdio.h>

```
main()
{
const int i=4;
float j;
j = ++i;
printf("%d %f", i,++j);
}
```

Answer:

Compiler error

22. main()

```
{
    int i=5,j=6,z;
    printf("%d",i+++j);
}
```

Answer:

11

23. main()

```
{
    int i =0;j=0;
    if(i && j++)
        printf("%d..%d",i++,j);
    printf("%d..%d,i,j);
}
```

Answer:

0..0

24. int i;

```
main(){
    int t;
    for ( t=4;scanf("%d",&i)-t;printf("%d\n",i))
        printf("%d--",t--);
}
```

// If the inputs are 0,1,2,3 find the o/p

Answer:

4--0

3--1

2--2

25. main(){

```
    int a= 0;int b = 20;char x =1;char y =10;
    if(a,b,x,y)
        printf("hello");
}
```

Answer:

hello

26. void main()

```
{
    unsigned giveit=-1;
    int gotit;
    printf("%u ",++giveit);
    printf("%u \n",gotit--giveit);
}
```

Answer:

0 65535

```

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

```

Answer:

I hate U

28. $a \ll 1$ is equivalent to
 a) multiplying by 2 b) dividing by 2 c) adding 2 d) none of the above
29. The operation of a stair case switch best explains the
 a) or operation b) and operation
 c) exclusive nor operation d) exclusive or operation
30. Which of the following is/are syntactically correct?
 a) for(); b) for(;); c) for(,); d) for(;;);
31. The expression $4 + 6/3 * 2 - 2 + 7\%3$ evaluates to
 a) 3 b) 4 c) 6 d) 7
32. Any C program
 a) must contain at least one function b) need not contain any function
 c) needs input data d) none of the above
33. Using goto inside for loop is equivalent to using
 a) continue b) break c) return d) none of the above
34. The program fragment

```
int a=5, b=2;
printf("%d", a+++++b);
```

 a) prints 7 b) prints 8 c) prints 9 d) none of the above
35. `printf("ab", "cd", "ef");` prints
 a) ab b) abcdef c) abcdef, followed by garbage value d) none of the above
36. Consider the following program segment.

```
i=6720; j=4;
while((i%j)==0)
{
    i=i/j;
    j=j+1;
}
```

 On termination j will have the value
 a) 4 b) 8 c) 9 d) 6720

UNIT-II

Predict the output or error(s) for the following:

```

37. 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:

```

    mmmm
    aaaa
    nnnn

```

```

38. main()
{
    extern int i;
    i=20;
    printf("%d",i);
}

```

Answer:

Linker Error : Undefined symbol '_i'

```

39. #define int char
main()
{
    int i=65;
    printf("sizeof(i)=%d",sizeof(i));
}

```

Answer:

```

    sizeof(i)=1

```

```

40. #define square(x) x*x
main()
{
    int i;
    i = 64/square(4);
    printf("%d",i);
}

```

Answer:

```

    64

```

```

41. #include <stdio.h>
#define a 10
main()
{
    #define a 50
    printf("%d",a);
}

```

```
}
```

Answer:
50

```
42. #define clrscr() 100
main()
{
    clrscr();
    printf("%d\n",clrscr());
}
```

Answer:
100

```
43. main()
{
    clrscr();
}
clrscr();
```

Answer:
No output/error

```
44. main()
{
    int i=1;
    while (i<=5)
    {
        printf("%d",i);
        if (i>2)
            goto here;
        i++;
    }
}
fun()
{
    here:
    printf("PP");
}
```

Answer:
Compiler error: Undefined label 'here' in function main

```
45. #define f(g,g2) g##g2
main()
{
    int var12=100;
    printf("%d",f(var,12));
}
```

Answer:
100

```

46. main()
{
    extern out;
    printf("%d", out);
}
int out=100;

```

Answer:
100

```

47. main()
{
    show();
}
void show()
{
    printf("I'm the greatest");
}

```

Answer:
Compiler error: Type mismatch in redeclaration of show.

```

48. int i,j;
    for(i=0;i<=10;i++)
    {
        j+=5;
        assert(i<5);
    }

```

Answer:
Runtime error: Abnormal program termination.
assert failed (i<5), <file name>,<line number>

```

49. #define FALSE -1
    #define TRUE 1
    #define NULL 0
    main() {
        if(NULL)
            puts("NULL");
        else if(FALSE)
            puts("TRUE");
        else
            puts("FALSE");
    }

```

Answer:
TRUE

```

50. #define max 5
    #define int arr1[max]
    main()
    {
        typedef char arr2[max];
        arr1 list={0,1,2,3,4};
    }

```

```
arr2 name="name";
printf("%d %s",list[0],name);
}
```

Answer:

Compiler error (in the line arr1 list = {0,1,2,3,4})

```
51. int i=10;
main()
{
extern int i;
{
int i=20;
{
const volatile unsigned i=30;
printf("%d",i);
}
printf("%d",i);
}
printf("%d",i);
}
```

Answer:

30,20,10

```
52. #include<stdio.h>
main()
{
int a[2][2][2] = { {10,2,3,4}, {5,6,7,8} };
int *p,*q;
p=&a[2][2][2];
*q=***a;
printf("%d..%d",*p,*q);
}
```

Answer:

garbagevalue..1

```
53. #include<stdio.h>
main()
{
register i=5;
char j[]="hello";
printf("%s %d",j,i);
}
```

Answer:

hello 5

```
54. main()
{
int i=_1_abc(10);
```

```

printf("%d\n",--i);
}
int _l_abc(int i)
{
return(i++);
}

```

Answer:

9

55. main()

```

{
char c=' ',x,convert(z);
getc(c);
if((c>='a') && (c<='z'))
x=convert(c);
printf("%c",x);
}
convert(z)
{
return z-32;
}

```

Answer:

Compiler error

56. main()

```

{
int i;
i = abc();
printf("%d",i);
}
abc()
{
_AX = 1000;
}

```

Answer:

1000

57. What are the following notations of defining functions known as?

i. int abc(int a,float b)

```

{
/* some code */
}

```

ii. int abc(a,b)

```

int a; float b;
{
/* some code*/
}

```

Answer:

- i. ANSI C notation
- ii. Kernighan & Ritchie notation

```
58. void main()
{
    static int i=5;
    if(--i){
        main();
        printf("%d ",i);
    }
}
```

Answer:
0 0 0 0

```
59. void main()
{
    int k=ret(sizeof(float));
    printf("\n here value is %d",++k);
}
int ret(int ret)
{
    ret += 2.5;
    return(ret);
}
```

Answer:
Here value is 7

```
60. void main()
{
    char a[]="12345\0";
    int i=strlen(a);
    printf("here in 3 %d\n",++i);
}
```

Answer:
here in 3 6

```
61. void main()
{
    int i;
    char a[]="\0";
    if(printf("%s\n",a))
        printf("Ok here \n");
    else
        printf("Forget it\n");
}
```

Answer:
Ok here

62. main()

```
{  
clrscr();  
}  
clrscr();
```

Answer:

No output/error

63. main()

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

Answer:

5 4 3 2 1

64. C preprocessor

- a) takes care of conditional compilation b) takes care of macros
c) takes care of include files d) acts before compilations

65. A preprocessor command

- a) need not start on a new line b) need not start on the first column
c) has # as the first character d) comes before the first executable statement

66. The following program

```
main()  
{  
int a=4;  
change(a);  
printf("%d",a);  
}  
change(int a)  
{ printf("%d",++a); }      outputs  
a)5 5   b)4 5   c) 5 4   d)4 4
```

67. The output of the following program is

```
main()  
{  
    static int x[]={ 1, 2, 3, 4, 5, 6, 7, 8};  
    int i;  
    for(i=2;i<6;i++)  
        x[x[i]]=x[i];  
    for(i=0; i<8;i++)  
        printf("%d",x[i]);  
}
```

a) 1 2 3 3 5 5 7 8 b) 1 2 3 4 5 6 7 8
c) 8 7 6 5 4 3 2 1 d) 1 2 3 5 4 6 7 8


```

        ++q;    }
    for(j=0;j<5;j++){
        printf(" %d ",*p);
        ++p;    }
}

```

Answer:

2 2 2 2 2 3 4 6 5

```

78. main()
{
    char *p="hai friends",*p1;
    p1=p;
    while(*p!='\0') ++*p++;
    printf("%s %s",p,p1);
}

```

Answer:

ibj!gsjfoet

```

79. void main()
{
    char far *farther,*farthest;

    printf("%d..%d",sizeof(farther),sizeof(farthest));

}

```

Answer:

4..2

```

80. main()
{
    char *p;
    p="Hello";
    printf("%c\n",&*p);
}

```

Answer:

H

```

81. main()
{
    static char names[5][20]={"pascal","ada","cobol","fortran","perl"};
    int i;
    char *t;
    t=names[3];
    names[3]=names[4];
    names[4]=t;
    for (i=0;i<=4;i++)
        printf("%s",names[i]);
}

```

Answer:

Compiler error: Lvalue required in function main

```
82. #include<stdio.h>
main()
{
    char s[]={'a','b','c','\n','c','\0'};
    char *p,*str,*str1;
    p=&s[3];
    str=p;
    str1=s;
    printf("%d",++*p + ++*str1-32);
}
```

Answer:

M

```
83. main()
{
    int a[2][3][2] = {{{2,4},{7,8},{3,4}},{2,2},{2,3},{3,4}}};
    printf("%u %u %u %d \n",a,*a,**a,***a);
    printf("%u %u %u %d \n",a+1,*a+1,**a+1,***a+1);
}
```

Answer:

100, 100, 100, 2

114, 104, 102, 3

```
84. main()
{
    int a[ ] = {10,20,30,40,50},j,*p;
    for(j=0; j<5; j++)
    {
        printf("%d" ,*a);
        a++;
    }
    p = a;
    for(j=0; j<5; j++)
    {
        printf("%d " ,*p);
        p++;
    }
}
```

Answer:

Compiler error: lvalue required.

```
85. main()
{
```

```

static int a[ ] = {0,1,2,3,4};
int *p[ ] = {a,a+1,a+2,a+3,a+4};
int **ptr = p;
ptr++;
printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
*ptr++;
printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
*++ptr;
printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
++*ptr;
printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
}

```

Answer:

```

111
222
333
344

```

86. pointers are of

- a) integer data type
- b) character data type
- c) unsigned integer data type
- d) none of these

87. main()

```

{
void *vp;
char ch = 'g', *cp = "goofy";
int j = 20;
vp = &ch;
printf("%c", *(char *)vp);
vp = &j;
printf("%d", *(int *)vp);
vp = cp;
printf("%s", (char *)vp + 3);
}

```

Answer:

```

g20fy

```

88. main ()

```

{
static char *s[ ] = {"black", "white", "yellow", "violet"};
char **ptr[ ] = {s+3, s+2, s+1, s}, ***p;
p = ptr;
**++p;
printf("%s", *--*++p + 3);
}

```

Answer:

ck

```
89. main()
{
    int i, n;
    char *x = "girl";
    n = strlen(x);
    *x = x[n];
    for(i=0; i<n; ++i)
    {
        printf("%s\n",x);
        x++;
    }
}
```

Answer:

```
(blank space)
irl
rl
l
```

```
90. main()
{
    char *cptr,c;
    void *vptr,v;
    c=10; v=0;
    cptr=&c; vptr=&v;
    printf("%c%v",c,v);
}
```

Answer:

Compiler error (at line number 4): size of v is Unknown.

```
91. main()
{
    char *str1="abcd";
    char str2[]="abcd";
    printf("%d %d %d",sizeof(str1),sizeof(str2),sizeof("abcd"));
}
```

Answer:

```
2 5 5
```

```
92. main()
{
    int *j;
    {
        int i=10;
        j=&i;
    }
}
```

```
    printf("%d",*j);  
}
```

Answer:

10

93. void main()

```
{  
    int const * p=5;  
    printf("%d",++(*p));  
}
```

Answer:

Compiler error: Cannot modify a constant value.

94. main()

```
{  
    char *p;  
    int *q;  
    long *r;  
    p=q=r=0;  
    p++;  
    q++;  
    r++;  
    printf("%p...%p...%p",p,q,r);  
}
```

Answer:

0001...0002...0004

95. main(int argc, char **argv)

```
{  
    printf("enter the character");  
    getchar();  
    sum(argv[1],argv[2]);  
}  
sum(num1,num2)  
int num1,num2;  
{  
    return num1+num2;  
}
```

Answer:

Compiler error.

96. # include <stdio.h>

```
int one_d[]={1,2,3};  
main()  
{  
    int *ptr;  
    ptr=one_d;  
    ptr+=3;  
    printf("%d",*ptr);  
}
```

Answer:

garbage value

```
97. # include<stdio.h>
    aaa() { printf("hi"); }

    bbb(){ printf("hello"); }
    ccc(){ printf("bye"); }
    main()
    {
        int (*ptr[3])();
        ptr[0]=aaa; ptr[1]=bbb;
        ptr[2]=ccc; ptr[2]();
    }
```

Answer:

bye

98. In the following pgm add a stmt in the function fun such that the address of 'a' gets stored in 'j'.

```
main(){
    int *j;
    void fun(int **);
    fun(&j);
}
void fun(int **k) {
    int a =0;
    /* add a stmt here*/
}
```

Answer:

```
*k = &a
```

99. main()

```
{
    char *p;
    p="%d\n";
    p++;
    p++;
    printf(p-2,300);
}
```

Answer:

```
300
```

100. func(a,b)

```
int a,b;
{
    return( a= (a==b) );
}
main()
{
    int process(),func();
    printf("The value of process is %d !\n ",process(func,3,6));
}
process(pf,val1,val2)
int (*pf) ();
int val1,val2;
{
    return((*pf) (val1,val2));
}
```

Answer:

```
The value if process is 0 !
```

101. main()

```
{
    char *p;
```

```

    printf("%d %d ",sizeof(*p),sizeof(p));
}

```

Answer:

1 2

102. main()

```

{
    char string[]="Hello World";
    display(string);
}
void display(char *string)
{
    printf("%s",string);
}

```

Answer:

Compiler Error : Type mismatch in redeclaration of function display

103. #include<stdio.h>

```

main()
{
    char s[]={'a','b','c','\n','c','\0'};
    char *p,*str,*str1;
    p=&s[3];
    str=p;
    str1=s;
    printf("%d",++*p + ++*str1-32);
}

```

Answer:

77

104. #include<stdio.h>

```

main()
{
    int a[2][2][2] = { {10,2,3,4}, {5,6,7,8} };
    int *p,*q;
    p=&a[2][2][2];
    *q=***a;
    printf("%d----%d",*p,*q);
}

```

Answer:

SomeGarbageValue---1

105. puts(argv[0]) prints

- a) the name of the source code file
- b) the number of command line arguments
- c) argv
- d) the name of the executable code file


```

{

printf("%d..%d..%d",BLACK,BLUE,GREEN);

return(1);
}

```

Answer:
0..1..2

112. #include<stdio.h>
main()
{
 struct xx
 {
 int x=3;
 char name[]="hello";
 };
 struct xx *s=malloc(sizeof(struct xx));
 printf("%d",s->x);
 printf("%s",s->name);
}

Answer:
Compiler Error

113. struct aaa{
 struct aaa *prev;
 int i;
 struct aaa *next;
};
main()
{
 struct aaa abc,def,ghi,jkl;
 int x=100;
 abc.i=0;abc.prev=&jkl;
 abc.next=&def;
 def.i=1;def.prev=&abc;def.next=&ghi;
 ghi.i=2;ghi.prev=&def;
 ghi.next=&jkl;
 jkl.i=3;jkl.prev=&ghi;jkl.next=&abc;
 x=abc.next->next->prev->next->i;
 printf("%d",x);
}

Answer:
2

114. struct point

```

{
int x;
int y;
};
struct point origin,*pp;
main()
{
pp=&origin;
printf("origin is(%d%d)\n",(*pp).x,(*pp).y);
printf("origin is (%d%d)\n",pp->x,pp->y);
}

```

Answer:

```

origin is(0,0)
origin is(0,0)

```

115. What is the output for the program given below

```

typedef enum errorType{warning, error, exception,}error;
main()
{
error g1;
g1=1;
printf("%d",g1);
}

```

Answer

Compiler error: Multiple declaration for error

116. typedef struct error{int warning, error, exception;}error;

```

main()
{
error g1;
g1.error =1;
printf("%d",g1.error);
}

```

Answer

1

117. main()

```

{
struct student
{
char name[30];
struct date dob;
}stud;
struct date
{
int day,month,year;
}
}

```

```

        };
        scanf("%s%d%d%d",stud.rollno,&student.dob.day,&student.dob.month,
&student.dob.year);
    }

```

Answer:

Compiler Error: Undefined structure date

118. Is the following code legal?

```

struct a
{
    int x;
    struct a *b;
}

```

Answer:

Yes.

119. #include<stdio.h>

```

main()
{
    struct xx
    {
        int x;
        struct yy
        {
            char s;
            struct xx *p;
        };
        struct yy *q;
    };
}

```

Answer:

Compiler Error

120. Structures may contain

a) multiple data items b) single data items c) a only d) a&b

121. The size of structure and union is same when they contain

a) single member b) any number of members c) a & b d) none

122. The operator used to find the size of any variable

a) sizeof b) Sizeof c) sizeOf d) all the above

123. The operator that is used to access the members of the structure using pointer variable

a) . b) -> c)* d) none of the above

124. The operator used to access the member of the structure

a) . b) -> c)* d) none of the above

125. The operator -> is same as the combination of the operators

a) * and . b) & and . c) * and & d) none of the above

fread reads 10 records and prints the names successfully. It will return EOF only when fread tries to read another record and fails reading EOF (and returning EOF). So it prints the last record again. After this only the condition feof(fp) becomes false, hence comes out of the while loop.

```
131. #define assert(cond) if(!(cond)) \
    (fprintf(stderr, "assertion failed: %s, file %s, line %d \n",#cond,\
    __FILE__,__LINE__), abort())
```

```
void main()
{
int i = 10;
if(i==0)
    assert(i < 100);
else
    printf("This statement becomes else for if in assert macro");
}
```

Answer:

No output

132. What is the problem with the following code segment?

```
while ((fgets(receiving array,50,file_ptr)) != EOF) ;
```

Answer:

fgets returns a pointer.

133. If a file is opened in r+ mode then

- | | |
|--|--------------------------|
| a) reading is possible | b) writing is possible |
| c) it will be created if it does not exist | d) appending is possible |

134. If a file is opened in w+ mode then

- | | |
|--|--------------------------|
| a) reading is possible | b) writing is possible |
| c) it will be created if it does not exist | d) appending is possible |

135. If a file is opened in r mode then

- | | |
|--|--------------------------|
| a) reading is possible | b) writing is possible |
| c) it will be created if it does not exist | d) appending is possible |

136. If a file is opened in a mode then

- | | |
|--|--------------------------|
| a) reading is possible | b) writing is possible |
| c) it will be created if it does not exist | d) appending is possible |

137. ftell

- | | |
|---|--|
| a) is a function | b) gives the current file position indicator |
| c) can be used to find the size of a file | d) none of the above |

138. The fseek function

- | | |
|----------------------|--------------------------------------|
| a) needs 2 arguments | b) makes rewind function unnecessary |
| c) takes 3 arguments | d) none of the above |

139. rewind function takes ___ number of arguments

- a) 1 b) 2 c) 3 d) 0

140. fseek(fp,0,0) is equivalent to

- | | | | |
|----------|-----------|----------|----------------------|
| a) ftell | b) rewind | c) a & b | d) none of the above |
|----------|-----------|----------|----------------------|

141. ferror function is used to find _____ errors
 a) logical b) file opening c) data d) all the above
142. The contents of the file are lost if it is opened in _____ mode
 a) a b) w c) w+ d) a+
143. The contents of the file are safe if it is opened in _____ mode
 a) a b) r c) a+b d) all the above
144. The valid binary modes of operation are
 a) ab b) rb+ c) wb+ d) ab+
145. rewind function is used to
 a) reset the file pointer b) point it to the end of the file
 c) stay at current position d) none of the above
146. feof function checks for
 a) file opening error b) data error c) end of file d) file closing error
147. The value returned by fopen() function when the file is not opened
 a) 0 b) garbage value c) NULL d) none of the above
148. The fcloseall() function performs
 a) closing of all the files b) closes all the files that are opened by that program
 c) closes only specified files d) none of the above
149. The function that is not used for random access to files is
 a) rewind b) ftell c) fseek d) fprintf

UNIT-VI

Predict the output or error(s) for the following:

```
150. main()
    {
    main();
    }
```

Answer:

Runtime error : Stack overflow.

151. The prefix equivalent for the postfix $ab+cd+*$ is
 a) $a+b*c+d$ b) $+ab*+cd$ c) $*+ab+cd$ d) $*++abcd$
152. The postfix equivalent for the prefix $*++abcd$ is
 a) $ab+c+d*$ b) $abcd++*$ c) $ab+cd+*$ d) $ab+c*d+$
153. The infix equivalent to the postfix expression $abc+d-*e\%f/$ is
 a) $a+b*c-d\%f/f$ b) $a*(b+c-d)\%e/f$ c) $a*b+c-d\%e/f$ d) $a*(b-c+d)\%e/f$
154. Evaluate the expression $2*3/5+6-4$
 a) 1 b) 2 c) 3 d) 4
155. The value of the prefix expression $+/*2-5\ 6\ 4\ 3$ is
 a) 1 b) 2 c) 3 d) 4
156. The value of the postfix expression $1\ 4\ +3\ /2\ *\ 6\ 4\ \% -$ is
 a) 1 b) -1 c) 0 d) 4
157. Towers of Hanoi is an application of
 a) stack b) queue c) linked list d) dequeue
158. The data structure used in railway reservation is
 a) stacks b) queues c) priority queues d) binary tree

159. The data structure applicable for a fully packed bus is
 a) stacks b) queues c) priority queues d) binary tree
160. The recursive functions are evaluated using
 a) stacks b) queues c) priority queues d) binary tree
161. The nested loops are evaluated using
 a) stacks b) queues c) structures d) binary tree
162. The data structure used in resource sharing systems is
 a) stacks b) queues c) arrays d) binary tree
163. Which of the following is not a linear data structure
 a) stacks b) queues c) linked list d) binary tree
164. In evaluation of postfix expression the data structure used is
 a) stacks b) queues c) arrays d) binary tree

UNIT-VII

165. Linked list uses _____ type of memory allocation
 a) static b) random c) dynamic d) compile time
166. Binary tree can be implemented using
 a) arrays b) double linked list c) a& b d) b only
167. In a complete binary tree, if the parent is at nth position then the children will be at
 a) $n+1, n+2$ b) $2n, 2n-1$ c) $2n, 2n+1$ d) $2n+1, 2n-1$
168. The number of non leaf nodes in a complete binary tree of height 5 is
 a) 16 b) 32 c) 31 d) 15
169. The number of leaf nodes in a complete binary tree of height 5 is
 a) 16 b) 32 c) 31 d) 15
170. The number of nodes in a complete binary tree of height 5 is
 a) 16 b) 32 c) 31 d) 15
171. The number of edges in a minimum cost spanning tree of n nodes is
 a) n b) $n+1$ c) $n-1$ d) $2n$
172. Traveling sales man problem is an application of
 a) spanning trees b) binary tree c) greedy method d) divide and conquer
173. The number of extra pointers required to reverse a singly linked list is
 a) 1 b) 2 c) 3 d) 4
174. The number of extra pointers required to reverse a double linked list is
 a) 1 b) 2 c) 3 d) 4
175. The functions used for memory allocation
 a) malloc b) calloc c) a&b d) none of the above
176. Linked lists uses _____ type of structures.
 a) nested structures b) self referential structures
 c) simple structures d) unions
177. _____ cannot be used to represent Linked lists.
 a) arrays b) structures c) unions d) all the above
178. Binary trees cannot be implemented using
 a) arrays b) unions c) single linked list d) all the above

179. `calloc(m,n)` is equivalent to
 a) `malloc(m*n,0)` b) `memset(0,m*n)`
 c) `ptr=malloc(m*n)` d) `malloc(m/n)`
180. Prim's and Kruskal's algorithms are used for finding solution to
 a) BFS b) DFS c) traveling salesman problem d) none of the above

UNIT-VIII

181. The time complexity of binary search in average case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
182. The time complexity of bubble sort in best case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
183. The time complexity of selection sort in best case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
184. The time complexity of insertion sort in best case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
185. The time complexity of quick sort in best case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
186. The time complexity of heap sort in best case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
187. The time complexity of merge sort in best case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
188. The best sorting technique among the following is
 a) quick b) heap c) merge d) bubble
189. In worst case quick sort behaves like
 a) insertion b) heap c) selection d) bubble
190. The time complexity of bubble sort in worst case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
191. The time complexity of selection sort in worst case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
192. The time complexity of insertion sort in worst case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
193. The time complexity of quick sort in worst case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
194. The time complexity of heap sort in worst case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
195. The time complexity of merge sort in worst case is
 a) $O(n)$ b) $O(n^2)$ c) $O(n \log n)$ d) $O(\log n)$
196. Quick sort is an application of
 a) partition exchange sort b) partition sort
 c) greedy method d) divide and conquer
197. Merge sort is an application of
 a) greedy method b) divide and conquer c) a&b d) none
198. The space complexity of Quick sort in average case is
 a) 0 b) $O(n)$ c) $O(n \log n)$ d) $O(\log n)$
199. The space complexity of bubble sort in worst case is
 a) 0 b) $O(n)$ c) $O(n \log n)$ d) $O(\log n)$

200. Binary search is effective only when the elements are in
a) ascending order b) descending order c) a& b d) jumbled order