1. What will be the output of the following program:

```
int main()
{
      int i;
      float a[5];
      for (i=0; i<5; i++)
          a[i] = (printf, ("%d",i/10.0));
      for (i=0; i<5; i++)
      printf("%.1f ",a[i]);
}
```

(a) Compile-Time Error (b) 0.0 0.0 0.0 0.0 0.0
(c) 0.0 0.1 0.2 0.3 0.4 (d) 1.0 1.0 1.0 1.0 1.0

2. What will be the output of the following program :

```
void func()
{
        printf("Testing...Done\n");
}
int main()
{
      func;
      func();
}
```

(a) Compile-Time Error (b) Testing...Done
(c) Testing...Done (d) None of these

3. A signed int bitfield 1-bit wide can only hold the values

(a) 0 and 1 (b) 0 and -1
(c) 0, 1 and -1 (d) None of these

4. What will be the output of the following program :

```
int main()
{
      int a=19,b=4;
      float c;
      c=a/b;
      printf("%f",c);
}
```

(a) 4.75 (b) 4
(c) 4.750000 (d) 4.000000

5. What will be the output of the following program :

```
int main()
{
   int _;
   _=70;
   printf("%d",_);
}
```

(a) Compile-Time Error (b) Run-Time Error
(c) 70 (d) None of these

6. In DOS environment, what is the maximum combined length of the command-line arguments passed to main (including the space between adjacent arguments and the name of the program itself).

(a) 80 Characters (b) 128 Characters

(c) Until RETURN KEY(i.e '\n') is encountered. (d) None of these

7. What will be the output of the following program :

```
int main()
{
   int (*foo)(char *, ...) = printf;
   (*foo)("hello, %s", "world!");
}
```

(a) Compile-Time error (b) hello, world!
(c) Run-Time Error (d) None of these

8. What will be the output of the following program :

```
int main()
{
   int i=5,(*foo)(char *, ...);
   foo=printf;
   printf("%d",i=(*foo)("hello, %s\n", "world!"));
}
```

(a) Compile-Time error (b) hello, world!
(c) hello, world! (d) hello, world!

9. What will be the output of the following program :

```
int main()
{
  int choice=2;
  switch(choice)
  {
     default:
          printf("Default1");

     case 1:
          printf("Case1");
          break;

     default:
          printf("Default2");
  }
}
```

(a) Compile-Time Error (b) Default1Case1
(c) Default2 (d) Default1

10. What is the MAXIMUM LIMIT for cases in a switch statement?

(a) 32767 cases (b) 257 cases
(c) 127 (d) None of these

11. What will be the output of the following program :

```
#define big(a,b) a > b ? a : b
#define swap(a,b) temp=a; a=b; b=temp;
int main()
{
   int a=3,b=5,temp;
   if ((3+big(a,b)) > b)
      swap(a,b);
   printf("%d %d",a,b);
}
```

(a) 3 0 (b) 5 3
(c) 3 5 (d) 5 0

12. What will be the output of the following program :

```
#define main main()
int main
{
   #define END }
   printf("First"
          "Second"
          "Third");
END
```

(a) Compile-Time Error (b) First
(c) FirstSecondThird (d) None of these (e) Second,Third

13. What will be the output of the following program :

```
int main()
{
  long double val;
  printf("%d bytes",sizeof(val));
}
```

(a) Compile-Time Error (b) 4 bytes
(c) 8 bytes (d) 10 bytes

14. What will be the output of the following program :

```
int * func()
{
      int temp=50;
      return &temp;
}
int main()
{
      int *val;
      val = func();
      printf("%d",*val);
}
```

(a) Compile-Time Error (b) 50
(c) Garbage Value (d) None of these

15. What will be the output of the following program :

```
struct {
      int i;
      float f;
}var;
int main()
{
      var.i=5;
      var.f=9.76723;
      printf("%d %.2f",var.i,var.f);
}
```

(a) Compile-Time Error (b) 5 9.76723
(c) 5 9.76 (d) 5 9.77

16. What will be the output of the following program :

```
struct {
      int i;
      float f;
};
int main()
{
      int i=5;
      float f=9.76723;
      printf("%d %.2f",i,f);
}
```

(a) Compile-Time Error (b) 5 9.76723
(c) 5 9.76 (d) 5 9.77

17. What will be the output of the following program :

```
struct values
{
       int i;
       float f;
};
int main()
{
      struct values var={555,67.05501};
      printf("%2d %.2f",var.i,var.f);
}
```

(a) Compile-Time Error (b) 55 67.05
(c) 555 67.06 (d) 555 67.05

18. What will be the output of the following program :

```
typedef struct {
      int i;
      float f;
}values;
int main()
{
      static values var={555,67.05501};
      printf("%2d %.2f",var.i,var.f);
}
```

(a) Compile-Time Error (b) 55 67.05
(c) 555 67.06 (d) 555 67.05

19. What will be the output of the following program :

```
struct my_struct{
      int i=7;
      float f=999.99;
}var;
int main()
{
  var.i=5;
  printf("%d %.2f",var.i,var.f);
}
```

(a) Compile-Time Error (b) 7 999.99
(c) 5 999.99 (d) None of these

20. What will be the output of the following program :

```
struct first{
      int a;
      float b;
}s1 = {32760,12345.12345};
typedef struct{
       char a;
       int b;
}second;
struct my_struct{
       float a;
       unsigned int b;
};
typedef struct my_struct third;
int main()
{
      static second s2={'A',- -4};
      third s3;
      s3.a=~(s1.a-32760);
      s3.b=-++s2.b;
      printf("%d %.2f\n%c %d\n%.2f %u",(s1.a)--
      ,s1.b+0.005,s2.a+32,s2.b,++(s3.a),--s3.b);
}
```

(a) Compile-Time Error (b) 32760 12345.12
(c) 32760 12345.13 (d) 32760 12345.13

21. What will be the output of the following program :

```
struct {
      int i,val[25];
}var={1,2,3,4,5,6,7,8,9},*vptr=&var;
int main()
{
      printf("%d %d %d\n",var.i,vptr->i,(*vptr).i);
      printf("%d %d %d %d %d %d",var.val[4],*(var.val+4),
                  vptr->val[4],
                  *(vptr-val+4),(*vptr).val[4],*((*vptr).val+4));
}
```

(a) Compile-Time Error (b) 1 1 1
(c) 1 1 1 (d) None of these

22. What will be the output of the following program :

```
typedef struct {
      int i;
      float f;
}temp;
void alter(temp *ptr,int x,float y)
{
      ptr->i=x;
      ptr->f=y;
}
int main()
{
      temp a={111,777.007};
      printf("%d %.2f\n",a.i,a.f);
      alter(&a,222,666.006);
      printf("%d %.2f",a.i,a.f);
}
```

(a) Compile-Time error (b) 111 777.007
(c) 111 777.01 (d) None of these

23. What will be the output of the following program :

```
typedef struct {
      int i;
      float f;
}temp;
temp alter(temp tmp,int x,float y)
{
      tmp.i=x;
      tmp.f=y;
      return tmp;
}
int main()
{
      temp a={111,777.007};
      printf("%d %.3f\n",a.i,a.f);
      a=alter(a,222,666.006);
      printf("%d %.3f",a.i,a.f);
}
```

(a) Compile-Time error (b) 111 777.007
(c) 111 777.01 (d) None of these

24. What will be the output of the following program :

```
typedef struct {
      int i;
      float f;
}temp;

temp alter(temp *ptr,int x,float y)
{
      temp tmp=*ptr;
      printf("%d %.2f\n",tmp.i,tmp.f);
      tmp.i=x;
      tmp.f=y;
      return tmp;
}
int main()
{
      temp a={65535,777.777};
      a=alter(&a,-1,666.666);
      printf("%d %.2f",a.i,a.f);
}
```

(a) Compile-Time error (b) 65535 777.777
(c) 65535 777.78 (d) -1 777.78

25. What will be the output of the following program :

```
struct my_struct1{
      int arr[2][2];
};
typedef struct my_struct1 record;
struct my_struct2{
      record temp;
}list[2]={1,2,3,4,5,6,7,8};
int main()
{
      int i,j,k;
      for (i=1; i>=0; i--)
            for (j=0; j<2; j++)
                  for (k=1; k>=0; k--)
                        printf("%d",list[i].temp.arr[j][k]);
}
```

(a) Compile-Time Error (b) Run-Time Error
(c) 65872143 (d) 56781243

26. What will be the output of the following program :

```
struct my_struct{
      int i;
      unsigned int j;
};
int main()
{
      struct my_struct temp1={-32769,-1},temp2;
      temp2=temp1;
      printf("%d %u",temp2.i,temp2.j);
}
```

(a) 32767 -1 (b) -32769 -1
(c) -32769 65535 (d) 32767 65535

27. What will be the output of the following program :

```
struct names {
      char str[25];
      struct names *next;
};
typedef struct names slist;
int main()
{
      slist *list,*temp;
      list=(slist *)malloc(sizeof(slist)); // Dynamic Memory
                                              Allocation
      strcpy(list->str,"Hai");
      list->next=NULL;
      temp=(slist *)malloc(sizeof(slist)); // Dynamic Memory
                                              Allocation
      strcpy(temp->str,"Friends");
      temp->next=list;
      list=temp;

      while (temp != NULL)
      {
            printf("%s",temp->str);
            temp=temp->next;
      }
}
```

(a) Compile-Time Error (b) HaiFriends
(c) FriendsHai (d) None of these

28. Which of the following declarations is NOT Valid :

(i)
```
struct A{
    int a;
    struct B{
       int b;
       struct B *next;
    }tempB;
    struct A *next;
}tempA;
```

(ii)
```
struct B{
    int b;
    struct B *next;
};
struct A{
    int a;
    struct B tempB;
    struct A *next;
};
```

(iii)
```
struct B{
      int b;
 }tempB;
 struct{
      int a;
      struct B *nextB;
 };
```

(iv)
```
struct B {
    int b;
    struct B{
       int b;
       struct B *nextB;
    }tempB;
    struct B *nextB;
}tempB;
```

(a) (iv) Only (b) (iii) Only
(c) All of the these (d) None of these

29. What will be the output of the following program :

```
union A{
      char ch;
      int i;
      float f;
}tempA;
int main()
{
      tempA.ch='A';
      tempA.i=777;
      tempA.f=12345.12345;
      printf("%d",tempA.i);
}
```

(a) Compile-Time Error (b) 12345
(c) Erroneous output (d) 777

30. What will be the output of the following program :

```
struct A{
      int i;
      float f;
      union B{
            char ch;
            int j;
      }temp;
}temp1;
int main()
{
      struct A temp2[5];
      printf("%d %d",sizeof temp1,sizeof(temp2));
}
```

(a) 6 30 (b) 8 40
(c) 9 45 (d) None of these

31. What will be the output of the following program :

```
int main()
{
      static struct my_struct{
            unsigned a:1;
            unsigned b:2;
            unsigned c:3;
            unsigned d:4;
            unsigned  :6; // Fill out first word
      } v = {1,2,7,12};
      printf("%d %d %d %d",v.a,v.b,v.c,v.d);
      printf("\nSize=%d bytes",sizeof v);
}
```

(a) Compile-Time Error (b) 1 2 7 12
(c) 1 2 7 12 (d) None of these

32. What will be the output of the following program :

```
int main()
{
      struct sample{
              unsigned a:1;
              unsigned b:4;
      }v={0,15};
      unsigned *vptr=&v.b;
      printf("%d %d",v.b,*vptr);
}
```

(a) Compile-Time Error (b) 0 0
(c) 15 15 (d) None of these

33. What will be the output of the following program :

```
int main()
{
      static struct my_struct{
            unsigned a:1;
            int i;
            unsigned b:4;
            unsigned c:10;
      }v={1,10000,15,555};
      printf("%d %d %d %d",v.i,v.a,v.b,v.c);
      printf("\nSize=%d bytes",sizeof v);
}
```

(a) Compile-Time Error (b) 1 10000 15 555
(c) 10000 1 15 555 (d) 10000 1 15 555

34. What will be the output of the following program :

```
int main()
{
      int val=1234;
      int* ptr=&val;
      printf("%d %d",++val,*ptr);
}
```

(a) 1234 1234 (b) 1235 1235
(c) 1234 1235 (d) 1235 1234

35. What will be the output of the following program :

```
int main()
{
      int val=1234;
      int* ptr=&val;
      printf("%d %d",val,*ptr++);
}
```

(a) 1234 1234 (b) 1235 1235

(c) 1234 1235 (d) 1235 1234

36. What will be the output of the following program :

```
int main()
{
      int val=1234;
      int *ptr=&val;
      printf("%d %d",val,++*ptr);
}
```

(a) 1234 1234 (b) 1235 1235
(c) 1234 1235 (d) 1235 1234

37. What will be the output of the following program :

```
int main()
{
      int val=1234;
      int *ptr=&val;
      printf("%d %d",val,(*ptr)++);
}
```

(a) 1234 1234 (b) 1235 1235
(c) 1234 1235 (d) 1235 1234

38. What will be the output of the following program :

```
int main()
{
      int val=1234;
      int *ptr=&val;
      printf("%d %d",++val,(*(int *)ptr)--);
}
```

(a) 1234 1233 (b) 1235 1234
(c) 1234 1234 (d) None of these

39. What will be the output of the following program :

```
int main()
{
      int a=555,*ptr=&a,b=*ptr;
      printf("%d %d %d",++a,--b,*ptr++);
}
```

(a) Compile-Time Error (b) 555 554 555
(c) 556 554 555 (d) 557 554 555

40. What will be the output of the following program :

```
int main()
{
      int a=555,b=*ptr,*ptr=&a;
      printf("%d %d %d",++a,--b,*ptr++);
}
```

(a) Compile-Time Error (b) 555 554 555
(c) 556 554 555 (d) 557 554 555

41. What will be the output of the following program :

```
int main()
{
      int a=555,*ptr=&a,b=*ptr;
      printf("%d %d %d",a,--*&b,*ptr++);
}
```

(a) Compile-Time Error (b) 555 554 555
(c) 556 554 555 (d) 557 554 555

42. What will be the output of the following program :

```
int main()
{
      int a=555,*ptr=&a,b=*ptr=777;
      printf("%d %d",--*&b,*(int *)&b);
}
```

(a) Compile-Time Error (b) 776 777
(c) 554 555 (d) None of these

43. What will be the output of the following program :

```
int main()
{
      int a=5u,*b,**c,***d,****e;
      b=&a;
      c=&b;
      d=&c;
      e=&d;
      printf("%u %u %u %u",*b-5,**c-11,***d-6,65535+****e);
}
```

(a) Compile-Time Error (b) 0 65530 65535 4
(c) 0 65530 65535 65539 (d) 0 -6 -1 -2

44. What will be the output of the following program :

```
int main()
{
      float val=5.75;
      int *ptr=&val;
      printf("%.2f %.2f",*(float *)ptr,val);
}
```

(a) Compile-Time Error (b) 5.75 5.75
(c) 5.00 5.75 (d) None of these

45. What will be the output of the following program :

```
int main()
{
      int val=50;
      const int *ptr1=&val;
      int const *ptr2=ptr1;
      printf("%d %d %d",++val,*ptr1,*ptr2);
      *(int *)ptr1=98;
      printf("\n%d %d %d",++val,*ptr1,*ptr2);
}
```

(a) Compile-Time Error (b) 51 50 50
(c) Run-Time Error (d) None of these

46. What will be the output of the following program :

```
int main()
{
      int val=77;
      const int *ptr1=&val;
      int const *ptr2=ptr1;
      printf("%d %d %d",--val,(*ptr1)++,*ptr2);
}
```

(a) Compile-Time Error (b) 77 78 77
(c) 76 77 77 (d) 77 77 77

47. What will be the output of the following program :

```
int main()
{
      int a=50,b=60;
      int* const ptr1=&a;
      printf("%d %d",--a,(*ptr1)++);
      ptr1=&b;
      printf("\n%d %d",++b,(*ptr1)++);
}
```

(a) Compile-Time Error (b) 49 50
(c) 50 50 (d) None of these

48. What will be the output of the following program :

```
int main()
{
      int a=50;
      const int* const ptr=&a;
      printf("%d %d",*ptr++,(*ptr)++);
}
```

(a) Compile-Time Error (b) 51 51
(c) 51 50 (d) None of these

49. What will be the output of the following program :

```
int main()
{
      int val=77;
      const int const *ptr=&val;
      printf("%d",*ptr);
}
```

(a) Compile-Time Error (b) Run-Time Error
(c) 77 (d) None of these

50. What will be the output of the following program :

```
int main()
{
      int a[]={1,2,3,4,5,6};
      int *ptr=a+2;
      printf("%d %d",--*ptr+1,1+*--ptr);
}
```

(a) Compile-Time Error (b) 1 2
(c) 2 3 (d) 1 3

51. What will be the output of the following program :

```
int main()
{
      int a[]={1,2,3,4,5,6};
      int *ptr=a+2;
      printf("%d %d",*++a,--*ptr);
}
```

(a) Compile-Time Error (b) 2 2
(c) 3 2 (d) 4 2

52. What will be the output of the following program :

```
int main()
{
        int matrix[2][3]={{1,2,3},{4,5,6}};
        printf("%d %d
                %d\n",*(*(matrix)),*(*(matrix+1)+2),
                *(*matrix+1));
        printf("%d %d
                %d",*(matrix[0]+2),*(matrix[1]+1),
                *(*(matrix+1)));
}
```

(a) Compile-Time Error (b) 1 5 2
(c) 1 6 2 (d) 1 6 2

53. What will be the output of the following program :

```
int main()
{
      int (*a)[5];
      printf("%d %d",sizeof(*a),sizeof(a));
}
```

(a) Compile-Time Error (b) 2 5
(c) 5 2 (d) None of these

54. The following code is not well-written. What does the program do ?

```
int   main()
{
      int a=1,b=2,c=3,d=4;printf("%d %d",a,b);
      printf(" %d %d",c,d);
}
```

(a) Run-Time Error (b) Compile-Time Error
(c) 1 2 3 4 (d) None of these

55. What will be the output of the following program :

```
int main()
{
      int a=1,b=2,c=3;
      c=(--a,b++)-c;
      printf("%d %d %d",a,b,c);
}
```

(a) 0 3 -3 (b) Compile-Time Error
(c) 0 3 -1 (d) 0 3 0

56. What will be the output of the following program :

```
int main()
{
      int a=1,b=2,c=3,d=4,e;
      e=(a,a)+(b,c)+(c,d)-(d,b);
      printf("%d",e);
}
```

(a) Compile-Time Error (b) 10
(c) 6 (d) 2

57. What will be the output of the following program :

```
int main()
{
      float val=2.;
      printf("%.2",val);
}
```

(a) Compile-Time error (b) 2.00
(c) %.2 (d) 2.000000

58. What will be the output of the following program :

```
int main()
{
      int a=5;
      int b=6;;
      int c=a+b;;;
      printf("%d",c);;;;
}
```

(a) Compile-Time Error (b) Run-Time Error
(c) 11 (d) None of these

59. What will be the output of the following program :

```
int main()
{
      int i,j;
      for (i=1; i<=3; i++)
            for (j=1; j<3; j++)
            {
                  if (i == j)
                        continue;
                  if ((j % 3) > 1)
                        break;
                  printf("%d",i);
            }
}
```

60. What will be the output of the following program :

```
#define swap(a,b) temp=a; a=b; b=temp;
int main()
{
      static int a=5,b=6,temp;
      if (a > b)
            swap(a,b);
      printf("a=%d b=%d",a,b);`
}
```

(a) a=5 b=6 (b) a=6 b=5
(c) a=6 b=0 (d) None of these

61. What will be the output of the following program :

```
int main()
{
      unsigned int val=5;
      printf("%u %u",val,val-11);
}
```

(a) Compile-Time error (b) 5 -6
(c) 5 65530 (d) None of these

62. What will be the output of the following program :

```
int main()
{
      int x=4,y=3,z=2;
      *&z*=*&x**&y;
      printf("%d",z);
}
```

(a) Compile-Time error (b) Run-Time Error
(c) 24 (d) Unpredictable

63. What will be the output of the following program :

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

(a) Compile-Time Error (b) i=1680 j=1680
(c) i=629 j=6561 (d) i=1681 j=3024

64. What will be the output of the following program :

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

(a) Compile-Time Error (b) 10 9 8 7 6
(c) 9 8 7 6 6 (d) 10 8 7 6 6

65. What will be the output of the following program :

```
int main()
{
      unsigned ch='Y';
      printf("%d",sizeof ch);
}
```

(a) Compile-Time Error (b) 2
(c) 4 (d) 1

66. What will be the output of the following program :

```
int main()
{
      int a=5;
      printf("%d");
}
```

(a) Compile-Time Error (b) 5
(c) Unpredictable (d) No Output

67. What will be the output of the following program :

```
int main()
{
      int a=5,b=6;
      printf("%d");
}
```

(a) Compile-Time Error (b) 5
(c) 6 (d) Unpredictable

68. What will be the output of the following program :

```
int main()
{
      int a=5,b=6,c=7;
      printf("%d %d %d");
}
```

(a) Compile-Time Error (b) 5 6 7
(c) 7 6 5 (d) Unpredictable

69. What will be the output of the following program :

```
#define Swap if (a != b) { \
                  temp=a; \
               a = b; \
                   b = temp; \ //Swapping Done
                       }
int main()
{
      int a=10,b=5,temp;
      Swap;
      printf("a=%d a=%d",a,b);
}
```

(a) Compile-Time Error (b) a=5 b=10
(c) a=10 b=5 (d) None of these

70. What will be the output of the following program :

```
int main()
{
      int val=50;
      void *ptr;
      ptr=&val;
      printf("%d %d",val,*(int *)ptr);
}
```

(a) Compile-Time Error (b) 50 50
(c) 50 0 (d) None of these

71. What will be the output of the following program :

```
int main()
{
      int a=5,b=6;
      printf("%d %d %d",a,b,--a*++b);
}
```

(a) Compile-Time Error (b) 5 6 30
(c) 4 7 28 (d) None of these

72. What will be the output of the following program :

```
int main()
{
      int val=5;
      void *ptr;
      *(int *)ptr=5;
      val=ptr;
      printf("%d %d",*(int *)val,*(int *)ptr);
}
```

(a) Compile-Time Error (b) Unpredictable
(c) 5 5 (d) None of these

73. What will be the output of the following program :

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

(a) Compile-Time Error (b) 3
(c) -1 (d) 0

74. What will be the output of the following program :

```
int main()
{
      int i,n=10;
      for (i=1; i<n--; i+=2)
            printf("%d\n",n-i);
}
```

(a) 84 (b) 840
(c) 852 (d) 864

75. What will be the output of the following program :

```
int main()
{
      printf("Hi!");
      if (-1)
            printf("Bye");
}
```

(a) No Output (b) Hi!
(c) Bye (d) Hi!Bye

76. What will be the output of the following program :

```
int main()
{
      printf("Hi!");
      if (0 || -1)
            printf("Bye");
}
```

(a) No Output (b) Hi!
(c) Bye (d) Hi!Bye

77. What will be the output of the following program :

```
int main()
{
      printf("Hi!");
      if (!1)
            printf("Bye");
}
```

(a) Compile-Time error (b) Hi!
(c) Bye (d) Hi!Bye

78. What will be the output of the following program :

```
int main()
{
      printf("Hi!");
      if !(0)
            printf("Bye");
}
```

(a) Compile-Time error (b) Hi!
(c) Bye (d) Hi!Bye

79. What will be the output of the following program :

```
int main()
{
      printf("Hi!");
      if (-1+1+1+1-1-1-1+(-1)-(-1))
            printf("Bye");
}
```

(a) No Output (b) Hi!
(c) Bye (d) Hi!Bye

80. What will be the output of the following program :

```
int main()
{
      if (sizeof(int) && sizeof(float) && sizeof(float)/2-
            sizeof(int))
            printf("Testing");
      printf("OK");
}
```

(a) No Output (b) OK
(c) Testing (d) TestingOK

81. What will be the output of the following program :

```
int main()
{
      int a=1,b=2,c=3,d=4,e;
      if (e=(a & b | c ^ d))
            printf("%d",e);
}
```

(a) 0 (b) 7
(c) 3 (d) No Output

82. What will be the output of the following program :

```
int main()
{
      unsigned val=0xffff;
      if (~val)
            printf("%d",val);
      printf("%d",~val);
}
```

(a) Compile-Time error (b) -1
(c) 0 (d) -1 0

83. What will be the output of the following program :

```
int main()
{
      unsigned a=0xe75f,b=0x0EF4,c;
      c=(a|b);
      if ((c > a) && (c > b))
            printf("%x",c);
}
```

(a) No Output (b) 0xe75f
(c) 0xefff (d) None of these

84. What will be the output of the following program :

```
int main()
{
      unsigned val=0xabcd;
      if (val>>16 | val<<16)
      {
            printf("Success");
            return;
      }
      printf("Failure");
}
```

(a) No Output (b) Success
(c) Failure (d) SuccessFailure

85. What will be the output of the following program :

```
int main()
{
      unsigned x=0xf880,y=5,z;
      z=x<<y;
      printf("%#x %#x",z,x>>y-1);
}
```

(a) 1000 f87 (b) 8800 0xf88
(c) 1000 f88 (d) 0x1000 0xf88

86. What will be the output of the following program :

```
int main()
{
      register int a=5;
      int *b=&a;
      printf("%d %d",a,*b);
}
```

(a) Compile-Time error (b) Run-Time error
(c) 5 5 (d) Unpredictable

87. What will be the output of the following program :

```
auto int a=5;
int main()
{
      printf("%d",a);
}
```

(a) Compile-Time error (b) Run-Time error
(c) 5 (d) Unpredictable

88. What will be the output of the following program :

```
int main()
{
      auto int a=5;
      printf("%d",a);
}
```

(a) Compile-Time error (b) Run-Time error
(c) 5 (d) Unpredictable

89. What will be the output of the following program :

```
int main()
{
      int a=1,b=2,c=3,d=4;
      if (d > c)
            if (c > b)
                  printf("%d %d",d,c);
            else if (c > a)
                  printf("%d %d",c,d);
            if (c > a)
                  if (b < a)
                        printf("%d %d",c,a);
                  else if (b < c)
                        printf("%d %d",b,c);
}
```

(a) 4 3 3 4 (b) 4 3 3 2
(c) 4 32 3 (d) 4 33 1

90. What will be the output of the following program :

```
int main()
{
      int a=1,b=2,c=3,d=4;
      if (d > c)
            if (c > b)
                  printf("%d %d",d,c);
            if (c > a)
                  printf("%d %d",c,d);
            if (c > a)
                  if (b < a)
                        printf("%d %d",c,a);
            if (b < c)
                  printf("%d %d",b,c);
}
```

(a) 4 32 3 (b) 4 33 42 3
(c) 4 3 3 4 2 3 (d) None of these

91. What will be the output of the following program :

```
int main()
{
      int a=1;
      if (a == 2);
            printf("C Program");
}
```

(a) No Output (b) C Program
(c) Compile-Time Error

92. What will be the output of the following program :

```
int main()
{
      int a=1;
      if (a)
            printf("Test");
      else;
            printf("Again");
}
```

(a) Again (b) Test
(c) Compile-Time Error (d) TestAgain

93. What will be the output of the following program :

```
int main()
{
      int i=1;
      for (; i<4; i++);
            printf("%d\n",i);
}
```

(a) No Output (b) 1
(c) 4 (d) None of these

94. What will be the output of the following program :

```
int main()
{
      int a,b;
      for (a=0; a<10; a++);
           for (b=25; b>9; b-=3);
              printf("%d %d",a,b);
}
```

(a) Compile-Time error (b) 10 9
(c) 10 7 (d) None of these

95. What will be the output of the following program :

```
int main()
{
      Float i;
      for (i=0.1; i<0.4; i+=0.1)
            printf("%.1f",i);
}
```

(a) 0.10.20.3 (b) Compile-Time Error
(c) Run-Time Error (d) No Output

96. What will be the output of the following program :

```
int main()
{
      int i;
      for (i=-10; !i; i++);
            printf("%d",-i);
}
```

(a) 0 (b) Compile-Time Error
(c) 10 (d) No Output

97. What will be the output of the following program :

```
int main()
{
      int i=5;
      do;
            printf("%d",i--);
      while (i>0);
}
```

(a) 5 (b) 54321
(c) Compile-Time Error (d) None of these

98. What will be the output of the following program :

```
int main()
{
      int i;
      for (i=2,i+=2; i<=9; i+=2)
            printf("%d",i);
}
```

(a) Compile-Time error (b) 2468
(c) 468 (d) None of these

99. What will be the output of the following program :

```
int main()
{
      int i=3;
      for (i--; i<7; i=7)
            printf("%d",i++);
}
```

(a) No Output (b) 3456
(c) 23456 (d) None of these

100. What will be the output of the following program :

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

(a) No Output (b) 54321
(c) 4321 (d) None of these

101)

```
int main()
{
      void *v;
      int integer=2;
      int *i=&integer;
      v=i;
      printf("%d\n",(int*)*v);
}
```

102)

```
int main()
{
      int i=i++,j=j++,k=k++;
      printf("%d%d%d",i,j,k);
}
```

103)

```
int main()
{
      static int i=i++, j=j++, k=k++;
      printf("i = %d j = %d k = %d", i, j, k);
}
```

104)

```
int main()
{
      while(1){
                  if(printf("%d",printf("%d")))
                     break;
                  else
                     continue;
      }
}
```

105)
```
main()
{
      unsigned int i=10;
      while(i-->=0)
            printf("%u ",i);
}
```

106)
```
#include<conio.h>
main()
{
      int x,y=2,z,a;
      if(x=y%2) z=2;
      a=2;
      printf("%d %d ",z,x);
}
```

107)
```
      main()
{
      int a[10];
      printf("%d",*a+1-*a+3);
}
```

108)
```
#define prod(a,b) a*b
main()
{
      int x=3,y=4;
      printf("%d",prod(x+2,y-1));
}
```

109)
```
main()
{
      unsigned int i=65000;
      while(i++!=0);
      printf("%d",i);
}
```

110)
```
main()
{
      int i=0;
      while(+(+i--)!=0)
            i-=i++;
      printf("%d",i);
}
```

111)
```
main()
{
      float f=5,g=10;
      enum{i=10,j=20,k=50};
      printf("%d\n",++k);
      printf("%f\n",f<<2);
      printf("%lf\n",f%g);
      printf("%lf\n",fmod(f,g));
}
```

112)
```
main()
{
      int i=10;
      void pascal f(int,int,int);
      f(i++,i++,i++);
      printf(" %d",i);
}
void pascal f(integer :i,integer:j,integer :k)
{
      write(i,j,k);
}
```

113)
```
void pascal f(int i,int j,int k)
{
      printf("%d %d %d",i, j, k);
}
void cdecl f(int i,int j,int k)
{
      printf("%d %d %d",i, j, k);
}
main()
{
      int i=10;
      f(i++,i++,i++);
      printf(" %d\n",i);
      i=10;
      f(i++,i++,i++);
      printf(" %d",i);
}
```

114) What is the output of the program given below

```
      main()
          {
             signed char i=0;
             for(;i>=0;i++) ;
             printf("%d\n",i);
          }
```

115)
```
main()
{
   unsigned char i=0;
   for(;i>=0;i++) ;
   printf("%d\n",i);
}
```

116) What is the output for the program given below:

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

117)
```
typedef struct error{int warning, error, exception;}error;
main()
{
      error g1;
      g1.error =1;
      printf("%d",g1.error);
}
```

118)
```
#ifdef something
int some=0;
#endif

main()
{
      int thing = 0;
      printf("%d %d\n", some ,thing);
}
```

119)
```
#if something == 0
int some=0;
#endif

main()
{
      int thing = 0;
      printf("%d %d\n", some ,thing);
}
```

120) What is the output for the following program

```
main()
{
      int arr2D[3][3];
      printf("%d\n", ((arr2D==* arr2D)&&(* arr2D == arr2D[0])) );
}
```

121)
```
int main()
{
      if(~0 == (unsigned int)-1)
            printf("You can answer this if you know how\ values
      are represented in memory");
}
```

122)
```
int swap(int *a,int *b)
      {
            *a=*a+*b;*b=*a-*b;*a=*a-*b;
      }
      main()
      {
            int x=10,y=20;
            swap(&x,&y);
            printf("x= %d y = %d\n",x,y);
      }
```

123)
```
main()
{
      char *p = "ayqm";
      printf("%c",++*(p++));
}
```

124)
```
main()
      {
             int i=5;
             printf("%d",++i++);
      }
```

125)
```
main()
      {
            char *p = "ayqm";
            char c;
            c = ++*p++;
            printf("%c",c);
      }
```

126)
```
      int aaa() {printf("Hi");}
      int bbb(){printf("hello");}
      iny ccc(){printf("bye");}

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

127)
```
main()
{
      int i=5;
      printf("%d",i=++i ==6);
}
```

128)
```
main()
{
      char p[ ]="%d\n";
      p[1] = 'c';
      printf(p,65);
}
```

129) What does this declaration mean?

```
void ( * abc( int, void ( *def) () ) ) ();
```

130)
```
main()
{
      while (strcmp("some","some\0"))
            printf("Strings are not equal\n");
}
```

131)
```
main()
{
      char str1[] = {'s','o','m','e'};
      char str2[] = {'s','o','m','e','\0'};
      while (strcmp(str1,str2))
            printf("Strings are not equal\n");
}
```

132)
```
main()
{
      int i = 3;
      for (;i++=0;) printf("%d",i);
}
```

133)
```
int main()
{
      int *mptr, *cptr;
      mptr = (int*)malloc(sizeof(int));
      printf("%d",*mptr);
      int *cptr = (int*)calloc(sizeof(int),1);
      printf("%d",*cptr);
}
```

134)
```
int main()
{
      static int i;
      while(i<=10)
            (i>2)?i++:i--;
      printf("%d", i);
}
```

135)
```
static int i;
while(i<=10)
      (i>2)?i++:i--;
printf("%d", I)
```

136)
```
1. const char *a;
2. char* const a;
3. char const *a;
-Differentiate the above declarations.
```

137)
```
main()
{
      int i=5,j=10;
      i=i&=j&&10;
      printf("%d %d",i,j);
}
```

138)
```
main()
{
      int i=4,j=7;
      j = j || i++ && printf("YOU CAN");
      printf("%d %d", i, j);
}
```

139)
```
main()
{
      register int a=2;
      printf("Address of a = %d",&a);
      printf("Value of a   = %d",a);
}
```

140)
```
main()
{
      float i=1.5;
      switch(i)
      {
            case 1: printf("1");
            case 2: printf("2");
            default : printf("0");
      }
}
```

```
141)  main()
      {
            extern i;
            printf("%d\n",i);
            {
                  int i=20;
                  printf("%d\n",i);
            }
      }
```

```
142)  main()
      {
            int a=2,*f1,*f2;
            f1=f2=&a;
            *f2+=*f2+=a+=2.5;
            printf("\n%d %d %d",a,*f1,*f2);
      }
```

```
143)  main()
      {
            char *p="GOOD";
            char a[ ]="GOOD";
            printf("\n sizeof(p) = %d, sizeof(*p) = %d, strlen(p) =
                     %d", sizeof(p), sizeof(*p), strlen(p));
            printf("\n sizeof(a) = %d, strlen(a) = %d", sizeof(a),
            strlen(a));
      }
```

```
144)  #define DIM( array, type) sizeof(array)/sizeof(type)
      main()
      {
            int arr[10];
            printf("The dimension of the array is %d",
                        DIM(arr, int));
      }
```

```
145)  int DIM(int array[])
      {
            return sizeof(array)/sizeof(int );
      }
      main()
      {
            int arr[10];
            printf("The dimension of the array is %d", DIM(arr));
      }
```

146)
```
main()
{
      static int a[3][3]={1,2,3,4,5,6,7,8,9};
      int i,j;
      static *p[]={a,a+1,a+2};
      for(i=0;i<3;i++)
      {
            for(j=0;j<3;j++)
                  printf("%d\t%d\t%d\t%d\n",*(*(p+i)+j),
                  *(*(j+p)+i),*(*(i+p)+j),*(*(p+j)+i));
      }
}
```

147)
```
main()
{
      void swap();
      int x=10,y=8;
      swap(&x,&y);
      printf("x=%d y=%d",x,y);
}
void swap(int *a, int *b)
{
   *a ^= *b,  *b ^= *a, *a ^= *b;
}
```

148)
```
main()
{
      int i = 257;
      int *iPtr = &i;
      printf("%d%d", *((char*)iPtr), *((char*)iPtr+1));
}
```

149)
```
main()
{
      int i = 258;
      int *iPtr = &i;
      printf("%d%d", *((char*)iPtr), *((char*)iPtr+1));
}
```

150)
```
main()
{
      int i=300;
      char *ptr = &i;
      *++ptr=2;
      printf("%d",i);
}
```

151)
```
#include <stdio.h>
main()
{
      char * str = "hello";
      char * ptr = str;
      char least = 127;
      while (*ptr++)
            least = (*ptr<least ) ?*ptr :least;
      printf("%d",least);
}
```

152) Declare an array of N pointers to functions returning pointers to functions returning pointers to characters?

153)
```
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);
}
```

154)
```
main()
{
      struct date;
      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);
}
```

155) There were 10 records stored in "somefile.dat" but the following program printed 11 names. What went wrong?

```
int main()
{
      struct student
      {
            char name[30], rollno[6];
      }stud;
      FILE *fp = fopen("somefile.dat","r");
      while(!feof(fp))
       {
            fread(&stud, sizeof(stud), 1 , fp);
            puts(stud.name);
      }
}
```

156) Is there any difference between the two declarations,

1. int foo(int *arr[]) and
2. int foo(int *arr[2])

157) What is the subtle error in the following code segment?

```
void fun(int n, int arr[])
{
      int *p=0;
      int i=0;
      while(i++<n)
            p = &arr[i];
      *p = 0;
}
```

158) What is wrong with the following code?

```
int *foo()
{
      int *s = malloc(sizeof(int)100);
      assert(s != NULL);
      return s;
}
```

159) What is the hidden bug with the following statement?

```
      assert(val++ != 0);
```

160)

```
int main()
{
      int *i = 0x400;  // i points to the address 400
      *i = 0;    // set the value of memory location pointed by i;
}
```

161)
```
#define assert(cond) if(!(cond)) \
                          (fprintf(stderr,"assertion failed: %s,
                           file %s, line %d \n",#cond,\
                           __FILE__,__LINE__), abort())

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

162) Is the following code legal?
```
struct a
{
      int x;
      struct a b;
}
```

163) Is the following code legal?
```
struct a
{
      int x;
      struct a *b;
}
```

164) Is the following code legal?
```
typedef struct a
{
      int x;
      aType *b;
}aType
```

165) Is the following code legal?
```
typedef struct a aType;
struct a
{
      int x;
      aType *b;
};
```

166) Is the following code legal?

```
int main()
{
      typedef struct a aType;
      aType someVariable;
      struct a
      {
            int x;
            aType *b;
      };
}
```

167)

```
int main()
{
      printf("sizeof (void *) = %d \n", sizeof( void *));
      printf("sizeof (int *) = %d \n", sizeof(int *));
      printf("sizeof (double *)  = %d \n", sizeof(double *));
      printf("sizeof(struct unknown *) = %d \n", sizeof(struct
      unknown *));
}
```

168)

```
char inputString[100] = {0};
```

To get string input from the keyboard which one of the following is better?

1) gets(inputString)
2) fgets(inputString, sizeof(inputString), fp)

169) Which version do you prefer of the following two,

```
      1) printf("%s",str);    // or the more curt one
      2) printf(str);
```

170)

```
int main()
{
      int i=10, j=2;
      int *ip= &i, *jp = &j;
      int k = *ip/*jp;
      printf("%d",k);
}
```

171)

```
int main()
{
      char ch;
      for(ch=0;ch<=127;ch++)
            printf("%c   %d \n", ch, ch);
}
```

172) Is this code legal?

```
int *ptr;
ptr = (int *) 0x400;
```

173)
```
main()
{
      char a[4]="HELLO";
      printf("%s",a);
}
```

174)
```
main()
{
      char a[4]="HELL";
      printf("%s",a);
}
```

175)
```
main()
{
      int a=10,*j;
      void *k;
      j=k=&a;
      j++;
      k++;
      printf("\n %u %u ",j,k);
}
```

176)
```
main()
{
      extern int i;
      {
            int i=20;
            {
                  const volatile unsigned i=30; printf("%d",i);
            }
            printf("%d",i);
      }
      printf("%d",i);
}
int i;
```

177) Printf can be implemented by using _____ list.

178)
```
char *someFun()
{
      char *temp = "string constant";
      return temp;
}
int main()
{
      puts(someFun());
}
```

179)
```
char *someFun1()
{
      char temp[ ] = "string";
      return temp;
}
char *someFun2()
{
      char temp[ ] = {'s', 't','r','i','n','g'};
      return temp;
}
int main()
{
      puts(someFun1());
      puts(someFun2());
}
```

180)
```
main()
{
      char i=0;
      for(;i>=0;i++) ;
      printf("%d\n",i);
}
```

181) Is the following statement a declaration/definition. Find what does it mean?

```
      int (*x)[10];
```

182. What will be the output of the following program :

```
int main()
{
  int choice=3;
  switch(choice)
  {
     default:
        printf("Default");

     case 1:
        printf("Choice1");
        break;

     case 2:
        printf("Choice2");
        break;
  }
}
```

(a) No Output (b) Default
(c) DefaultChoice1 (d) None of these

183. What will be the output of the following program :

```
int main()
{
  static int choice;
  switch(--choice,choice-1,choice-1,choice+=2)
  {
     case 1:
        printf("Choice1");
        break;

     case 2:
        printf("Choice2");
        break;

     default:
        printf("Default");
  }
}
```

(a) Choice1 (b) Choice2
(c) Default (d) None of these

184. What will be the output of the following program :

```
int main()
{
      for (;printf(""););
}
```

(a) Compile-Time error (b) Executes ONLY once
(c) Executes INFINITELY (d) None of these

185. What will be the output of the following program :

```
int main()
{
      int i;
      for (;(i=4)?(i-4):i++;)
            printf("%d",i);
}
```

(a) Compile-Time error (b) 4
(c) Infinite Loop (d) No Output

186. What will be the output of the following program :

```
int main()
{
  static int j;
  for (j<5; j<5; j+=j<5)
      printf("%d",j++);
}
```

(a) 024 (b) Compile-Time Error
(c) 01234 (d) No Output

187. What will be the output of the following program :

```
int main()
{
  int i=9;
  for (i--; i--; i--)
      printf("%d ",i);
}
```

(a) 9 6 3 (b) Compile-Time Error
(c) 7 5 3 1 (d) Infinite Loop

188. What will be the output of the following program :

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

(a) 6 4 2 (b) Compile-Time Error
(c) 6 3 1 (d) Infinite Loop

189. Which of the following code causes INFINITE Loop :

(a) do while(1); (b) do;while(1);
(c) do; (d) do{}while(1);

(i) Only (a)
(ii) Only (b), (c) & (d)
(iii) None of these
(iv) All of these

190. What will be the output of the following program :

```
#define Loop(i) for (j=0; j<i; j++)
{ \
    sum += i+j;     \
}
int main()
{
      int i,j,sum=0;
      for (i=0; i<=3; i++)
      Loop(i)
                printf("%d",sum);
}
```

(a) Run-Time Error (b) Compile-Time Error
(c) 18 (d) 0

ANSWER KEY/EXPLANATION

1. Ans. (c) Since Comma operator (,) is used, the expressions are evaluated from left to right and the entire expression assumes the value of the last one evaluated. Also i/10.0 is the rightmost expression hence it is evaluated and its value is assigned to a[i]. The first printf in the program has no effect on the output (See next question).

2. Ans. (b) Since first statement 'func;' has no effect on the output and the second statement
'func();' is a normal function call which prints the message(One time) as specified.

3. Ans. (b) Since any non-zero value will be interpreted as -1.

4. Ans. (d) Since a and b are both of type int, so the result of a/b was of type int. That was
converted to type float when you assigned it to c, but the conversion took place after the
division, not before.

5. Ans. (c) Since a variable (an identifier) can start with an underscore.

6. Ans. (b) Since this is a DOS limit.

7. Ans. (b) This is one of the method used for calling functions through pointers. Here foo is a pointer to a function (printf) that accepts an argument which is a pointer to a character. Also it accepts variable number of arguments and returns an integer value.

8. Ans. (d) This is another method used for calling functions through pointers. Here foo is a pointer to a function (printf) that accepts an argument which is a pointer to a character. Also it accepts variable number of arguments and returns an integer value. Also 'printf' returns theno. of bytes output which is stored in the variable 'i' i.e. i=14 (including '\n' character).

9. Ans. (a) Since Too many default cases i.e. the compiler encountered more than one default statement in a single switch.

10. Ans (b)

11. Ans. (d) Since if ((3+3 > 5 ? 3 : 5) > 5) is FALSE. But a macro is expanded during the compilation process itself. Thus temp=a; will not be executed and the remaining statements a=b; and b=temp; are executed in the normal way.

12. Ans. (c) Preprocessor directives can be defined anywhere in the program but just before its use. C will automatically do the concatenation for you on very long strings, resulting in nicer looking programs. According to K&R (applies to ALL C compilers), a string constant consists of exactly one string unit, containing double quotes, text, double quotes ("like this"). You must use the backslash(\) as a continuation character in order to extend a string constant across line boundaries.
Thus the above printf statement could be re-written (K&R Style) as follows :

```
printf("First" \
       "Second" \
       "Third");
```

13. Ans. (c) Here the address of the variable 'temp' is returned to the variable 'val' in the main
program but the variable 'temp' is local to the function. As we know every function involves built-in stack and whenever a function is called Stack Pointer (SP, which is a register pseudo-variable) value changes.

14. Ans. (d) Though both <struct type name> and <structure variables> are optional, one of the two must appear. In the above program, <structure variable> i.e. var is used. (2 decimal places or) 2-digit precision of 9.76723 is 9.77

15. Ans. (d) Both <struct type name> and <structure variables> are optional. Thus the structure
defined in the above program has no use and program executes in the normal way.

16. Ans. (c) The members of a structure variable can be assigned initial values in much the same
manner as the elements of an array. The initial values must appear in order in which they will be assigned to their corresponding strucutre members, enclosed in braces and separated by commas.

17. Ans. (c) In the above program, values is the user-defined structure type or the new user-defined data type. Structure variables can then be defined in terms of the new data type.

18. Ans. (a) C language does not permit the initialization of individual structure members within the template. The initialization must be done only in the declaration of the actual variables. The correct way to initialize the values is shown in earlier answers.

19. Ans. (d) Illustrating 3 different ways of declaring the structres : first, second and third are the user-defined structure type. s1, s2 and s3 are structure variables. Also an expression of the form ++variable.member is equivalent to ++(variable.member), i.e. ++ operator will apply to the structure member, not the entire structure variable.

20. Ans. (b) Since value of the member 'i' can be accessed using var.i, vptr->i and (*vptr).i
Similarly 5th value of the member 'val' can be accessed using var.val[4], *(var.val+4), vptr->val[4], *(vptr->val+4), (*vptr).val[4] and *((*vptr).val+4).

21. Ans. (c) This program illustrates the transfer of a structure to a function by passing the structure's address (a pointer) to the function.

22. Ans. (b) This program illustrates the transfer of a structure to a function by value. Also the altered structure is now returned directly to the calling portion of the program.

23. Ans. (d) This program illustrates the transfer of a structure to a function by passing the structure's address (a pointer) to the function. Also the altered structure is now returned directly to the calling portion of the program.

24. Ans. (c) This program illustrates the implementation of a nested structure i.e. structure inside another structure.

25. Ans. (d) An entire structure variable can be assigned to another structure variable, provided both variables have the same composition.

26. Ans. (c) It is sometimes desirable to include within a structure one member i.e. a pointer to the parent structure type. Such structures are known as Self-Referencial structures. These structures are very useful in applications that involve linked data structures, such as lists and trees. [A linked data structure is not confined to some maximum number of components. Rather, the data structure can expand or contract in size as required.]

27. Ans. (d) Since all the above structure declarations are valid in C.

28. Ans. (c) The above program produces erroneous output (which is machine dependent). In effect,a union creates a storage location that can be used by any one of its members at a time. When a different member is assigned a new value, the new value supercedes the previous member's value.

[NOTE : The compiler allocates a piece of storage that is large enough to hold the largest
variable type in the union i.e. all members share the same address.]

29. Ans. (b) Since int (2 bytes) + float (4 bytes) = (6 bytes) + Largest among union is int (2 bytes) is equal to (8 bytes). Also the total number of bytes the array 'temp2' requires :
(8 bytes) * (5 bytes) = (40 bytes).

30. Ans. (b) The four fields within 'v' require a total of 10 bits and these bits can be accommodated within the first word(16 bits). Unnamed fields can be used to control the alignment of bit fields within a word of memory. Such fields provide padding within the word.
[NOTE : Some compilers order bit-fields from righ-to-left (i.e. from lower-order bits to high-order bits) within a word, whereas other compilers order the fields from left-to-right (high-order to low-order bits).

31. Ans. (c)a=1 (1 bit: 0 or 1)
b=3 (2 bits: 00 or 01 or 10 or 11),
c=7 (3 bits: 000 or 001 or 010 or 011 or 100 or 101 or 110 or 111)
d=15 (4 bits: 0000 or 0001 or 0010 or 0011 or 0100 or 0101 or 0110 or 0111 or 1000 or
1001 or 1010 or 1011 or 1100 or 1101 or 1110 or 1111)

32. Ans. (a) Since we cannot take the address of a bit field variable i.e. Use of pointer to access
the bit fields is prohibited. Also we cannot use 'scanf' function to read values into a bit field as it requires the address of a bit field variable. Also array of bit-fields are not permitted and a function cannot return a bit field.

33. Ans. (d) Here the bit field variable 'a' will be in first byte of one word, the variable 'i' will be in the second word and the bit fields 'b' and 'c' will be in the third word. The variables 'a', 'b' and 'c' would not get packed into the same word. [NOTE: one word=2 bytes]

34. Ans. (d)

35. Ans. (a)

36. Ans. (b)

37. Ans. (d)

38. Ans. (c)

39. Ans. (c)

40. Ans. (a)

41. Ans. (b)

42. Ans. (b)

43. Ans. (b)

44. Ans. (b)

45. Ans. (d)

46. Ans. (a)

47. Ans. (a)

48. Ans. (a)

49. Ans. (c)

50. Ans. (c)

51. Ans. (b)

52. Ans. (c)

53. Ans. (d)

54. Ans. (c)

55. Ans. (c)

56. Ans. (c)

57. Ans. (c)
58. Ans. (c)
59. Ans. 23
60. Ans. (c)
61. Ans. (c)
62. Ans. (c)
63. Ans. (c) Compiler dependent
64. Ans. (d)
65. Ans. (b)
66. Ans. (b)
67. Ans. (c)
68. Ans. (c)
69. Ans. (a)
70. Ans. (b)

71. Ans. (a)
72. Ans. (c)
73. Ans. (c)
74. Ans. (c)
75. Ans. (d)
76. Ans. (d)
77. Ans. (b)
78. Ans. (a)
79. Ans. (d)
80. Ans. (b)
81. Ans. (b)
82. Ans. (c)
83. Ans. (c)
84. Ans. (b)
85. Ans. (d)
86. Ans. (a)
87. Ans. (a)
88. Ans. (c)
89. Ans. (c)
90. Ans. (b)
91. Ans. (b)
92. Ans. (d)
93. Ans. (c)
94. Ans. (c)
95. Ans. (a)
96. Ans. (c)
97. Ans. (c)
98. Ans. (c)
99. Ans. (d)
100. Ans. (c)
101. **Answer:** Compliler Error: Can't dereference void Pointer directly.

Explanation:
Void pointer is a generic pointer type. No pointer arithmetic can be done on it. Void pointers are normally used for,

1. Passing generic pointers to functions and returning such pointers.
2. As a intermediate pointer type.
3. Used when the exact pointer type will be known at a later point of time.

102. Answer: Garbage values.

Explanation:
An identifier is available to use in program code from the point of its declaration.
So expressions such as i = i++ are valid statements. The i, j and k are automatic variables and so they contain some garbage value. Garbage in is garbage out (GIGO).

103. Answer: i = 1 j = 1 k = 1

Explanation:
Since static variables are initialized to zero by default.

104. Answer: Garbage values

Explanation:
The inner printf executes first to print some garbage value. The printf returns no of characters printed and this value also cannot be predicted. Still the outer printf prints something and so returns a non-zero value. So it encounters the break statement and comes out of the while statement.

105. Answer: 10 9 8 7 6 5 4 3 2 1 0 65535 65534…..

Explanation:
Since i is an unsigned integer it can never become negative. So the expression i-- >=0 will always be true, leading to an infinite loop.

106. Answer: Garbage-value 0

Explanation:
The value of y%2 is 0. This value is assigned to x. The condition reduces to if (x) or in other words if(0) and so z goes uninitialized.
Thumb Rule: Check all control paths to write bug free code.

107. Answer: 4

Explanation:
*a and -*a cancels out. The result is as simple as 1 + 3 = 4 !

108. Answer: 10

Explanation:
The macro expands and evaluates to as:
x+2*y-1 => x+(2*y)-1 => 10

109. Answer: 1

Explanation:
Note the semicolon after the while statement. When the value of i becomes 0 it comes out of while loop. Due to post-increment on i the value of i while printing is 1.

110. Answer: -1

Explanation:
Unary + is the only dummy operator in C. So it has no effect on the expression and now the while loop is,
while(i--!=0) which is false and so breaks out of while loop. The value –1 is printed due to the post-decrement operator.

111. Answer: Line no 5: Error: Lvalue required
Line no 6: Cannot apply leftshift to float
Line no 7: Cannot apply mod to float
Explanation:
Enumeration constants cannot be modified, so you cannot apply ++.
Bit-wise operators and % operators cannot be applied on float values.
fmod() is to find the modulus values for floats as % operator is for ints.

112. Answer: Compiler error: unknown type integer
Compiler error: undeclared function write

Explanation:
Pascal keyword doesn't mean that pascal code can be used. It means that the function follows Pascal argument passing mechanism in calling the functions.

113. Answer: 10 11 12 13
12 11 10 13
Explanation:
Pascal argument passing mechanism forces the arguments to be called from left to right. cdecl is the normal C argument passing mechanism where the arguments are passed from right to left.

114. Answer -128
Explanation
Notice the semicolon at the end of the for loop. THe initial value of the i is set to 0. The inner loop executes to increment the value from 0 to 127 (the positive range of char) and then it rotates to the negative value of -128. The condition in the for loop fails and so comes out of the for loop. It prints the current value of i that is -128.

115. Answer infinite loop
Explanation
The difference between the previous question and this one is that the char is declared to be unsigned. So the i++ can never yield negative value and i>=0 never becomes false so that it can come out of the for loop.

116. Answer Compiler error: Multiple declaration for error

Explanation

The name error is used in the two meanings. One means that it is a enumerator constant with value 1. The another use is that it is a type name (due to typedef) for enum errorType. Given a situation the compiler cannot distinguish the meaning of error to know in what sense the error is used:

```
error g1;
g1=error;
// which error it refers in each case?
```

When the compiler can distinguish between usages then it will not issue error (in pure technical terms, names can only be overloaded in different namespaces).

Note: the extra comma in the declaration,

```
enum errorType{warning, error, exception,}
```

is not an error. An extra comma is valid and is provided just for programmer's convenience.

117. Answer: 1

Explanation

The three usages of name errors can be distinguishable by the compiler at any instance, so valid (they are in different namespaces).

```
Typedef struct error{int warning, error, exception;}error;
```

This error can be used only by preceding the error by struct kayword as in:

```
struct error someError;
typedef struct error{int warning, error, exception;}error;
```

This can be used only after . (dot) or -> (arrow) operator preceded by the variable name as in :

```
g1.error =1;
printf("%d",g1.error);
     typedef struct error{int warning, error,
exception;}error;
```

This can be used to define variables without using the preceding struct keyword as in:

```
error g1;
```

Since the compiler can perfectly distinguish between these three usages, it is perfectly legal and valid.

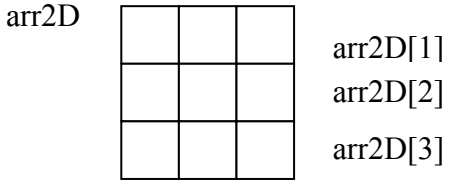
Note

This code is given here to just explain the concept behind. In real programming don't use such overloading of names. It reduces the readability of the code. Possible doesn't mean that we should use it!

118. **Answer:** Compiler error : undefined symbol some
Explanation:
This is a very simple example for conditional compilation. The name something is not already known to the compiler making the declaration
int some = 0;
effectively removed from the source code.

119. Answer: 0 0
Explanation
This code is to show that preprocessor expressions are not the same as the ordinary expressions. If a name is not known the preprocessor treats it to be equal to zero.

120. Answer
1
Explanation
This is due to the close relation between the arrays and pointers. N dimensional arrays are made up of (N-1) dimensional arrays.
arr2D is made up of a 3 single arrays that contains 3 integers each .



The name arr2D refers to the beginning of all the 3 arrays. *arr2D refers to the start of the first 1D array (of 3 integers) that is the same address as arr2D. So the expression (arr2D == *arr2D) is true (1).
Similarly, *arr2D is nothing but *(arr2D + 0), adding a zero doesn't change the value/meaning. Again arr2D[0] is the another way of telling *(arr2D + 0). So the expression (*(arr2D + 0) == arr2D[0]) is true (1).
Since both parts of the expression evaluates to true the result is true(1) and the same is printed.
121. Answer: You can answer this if you know how values are represented in memory
Explanation
~ (tilde operator or bit-wise negation operator) operates on 0 to produce all ones to fill the space for an integer. -1 is represented in unsigned value as all 1's and so both are equal.

122. Answer: x = 20 y = 10
Explanation
This is one way of swapping two values. Simple checking will help understand this.
123. Answer:
b

124. Answer: Compiler error: Lvalue required in function main

Explanation:

++i yields an rvalue. For postfix ++ to operate an lvalue is required.

125. Answer: b

Explanation:

There is no difference between the expression ++*(p++) and ++*p++. Parenthesis just works as a visual clue for the reader to see which expression is first evaluated.

126. Answer: bye

Explanation:

int (* ptr[3])() says that ptr is an array of pointers to functions that takes no arguments and returns the type int. By the assignment ptr[0] = aaa; it means that the first function pointer in the array is initialized with the address of the function aaa. Similarly, the other two array elements also get initialized with the addresses of the functions bbb and ccc. Since ptr[2] contains the address of the function ccc, the call to the function ptr[2]() is same as calling ccc(). So it results in printing "bye".

127. Answer: 1

Explanation: The expression can be treated as i = (++i==6), because == is of higher precedence than = operator. In the inner expression, ++i is equal to 6 yielding true(1). Hence the result.

128. Answer: A

Explanation:

Due to the assignment p[1] = 'c' the string becomes, "%c\n". Since this string becomes the format string for printf and ASCII value of 65 is 'A', the same gets printed.

129. Answer: abc is a ptr to a function which takes 2 parameters .(a). an integer variable.(b). a ptrto a funtion which returns void. the return type of the function is void.

Explanation: Apply the clock-wise rule to find the result.

130. Answer: No output

Explanation:

Ending the string constant with \0 explicitly makes no difference. So "some" and "some\0" are equivalent. So, strcmp returns 0 (false) hence breaking out of the while loop.

131. Answer: "Strings are not equal"
"Strings are not equal"

Explanation:
If a string constant is initialized explicitly with characters, '\0' is not appended automatically to the string. Since str1 doesn't have null termination, it treats whatever the values that are in the following positions as part of the string until it randomly reaches a '\0'. So str1 and str2 are not the same, hence the result.

132. Answer: Compiler Error: Lvalue required.
Explanation:
As we know that increment operators return rvalues and hence it cannot appear on the left hand side of an assignment operation.

133. Answer: garbage-value 0
Explanation:
The memory space allocated by malloc is uninitialized, whereas calloc returns the allocated memory space initialized to zeros.

134. Answer: 32767
Explanation:
Since i is static it is initialized to 0. Inside the while loop the conditional operator evaluates to false, executing i--. This continues till the integer value rotates to positive value (32767). The while condition becomes false and hence, comes out of the while loop, printing the i value.

135. Answer: 10 10
Explanation:
The Ternary operator (? :) is equivalent for if-then-else statement. So the question can be written as:

```
if(i,j)
     {
if(i,j)
     j = i;
else
    j = j;
}
  else
    j = j;
```

136. Answer:

1. 'const' applies to char * rather than 'a' (pointer to a constant char)

```
*a='F'      : illegal
a="Hi"      : legal
```

2. 'const' applies to 'a' rather than to the value of a (constant pointer to char)

```
*a='F'      : legal
a="Hi"      : illegal
```

4. Same as 1.

137. Answer: 1 10

Explanation:

The expression can be written as i=(i&=(j&&10)); The inner expression (j&&10) evaluates to 1 because j==10. i is 5. i = 5&1 is 1. Hence the result.

138. Answer: 4 1

Explanation:

The boolean expression needs to be evaluated only till the truth value of the expression is not known. j is not equal to zero itself means that the expression's truth value is 1. Because it is followed by || and true || (anything) => true where (anything) will not be evaluated. So the remaining expression is not evaluated and so the value of i remains the same.

Similarly when && operator is involved in an expression, when any of the operands become false, the whole expression's truth value becomes false and hence the remaining expression will not be evaluated.
false && (anything) => false where (anything) will not be evaluated.

139. Answer: Compier Error: '&' on register variable

Rule to Remember:

& (address of) operator cannot be applied on register variables.

140. Answer: Compiler Error: switch expression not integral

Explanation:

Switch statements can be applied only to integral types.

141. Answer: Linker Error : Unresolved external symbol i

Explanation:

The identifier i is available in the inner block and so using extern has no use in resolving it.

142. Answer: 16 16 16
Explanation:
f1 and f2 both refer to the same memory location a. So changes through f1 and f2 ultimately affects only the value of a.

143. Answer: sizeof(p) = 2, sizeof(*p) = 1, strlen(p) = 4
sizeof(a) = 5, strlen(a) = 4
Explanation:
sizeof(p) => sizeof(char*) => 2
sizeof(*p) => sizeof(char) => 1
Similarly,
sizeof(a) => size of the character array => 5
When sizeof operator is applied to an array it returns the sizeof the array and it is not the same as the sizeof the pointer variable. Here the sizeof(a) where a is the character array and the size of the array is 5 because the space necessary for the terminating NULL character should also be taken into account.

144. Answer: 10
Explanation:
The size of integer array of 10 elements is 10 * sizeof(int). The macro expands to sizeof(arr)/sizeof(int) => 10 * sizeof(int) / sizeof(int) => 10.

145. Answer: 1
Explanation:
Arrays cannot be passed to functions as arguments and only the pointers can be passed. So the argument is equivalent to int * array (this is one of the very few places where [] and * usage are equivalent). The return statement becomes, sizeof(int *)/ sizeof(int) that happens to be equal in this case.

146. Answer:

```
          1       1       1       1
          2       4       2       4
    3       7       3       7
          4       2       4       2
          5       5       5       5
    6       8       6       8
          7       3       7       3
          8       6       8       6
    9       9       9       9
```

Explanation:
((p+i)+j) is equivalent to p[i][j].

147. Answer: x=10 y=8
Explanation:
Using ^ like this is a way to swap two variables without using a temporary variable and that too in a single statement.
Inside main(), void swap(); means that swap is a function that may take any number of arguments (not no arguments) and returns nothing. So this doesn't issue a compiler error by the call swap(&x,&y); that has two arguments.
This convention is historically due to pre-ANSI style (referred to as Kernighan and Ritchie style) style of function declaration. In that style, the swap function will be defined as follows,

```
void swap()
int *a, int *b
{
   *a ^= *b,  *b ^= *a, *a ^= *b;
}
```

where the arguments follow the (). So naturally the declaration for swap will look like, void swap() which means the swap can take any number of arguments.

148. Answer: 1 1
Explanation:
The integer value 257 is stored in the memory as, 00000001 00000001, so the individual bytes are taken by casting it to char * and get printed.

149. Answer: 2 1
Explanation:
The integer value 257 can be represented in binary as, 00000001 00000001. Remember that the INTEL machines are 'small-endian' machines. Small-endian means that the lower order bytes are stored in the higher memory addresses and the higher order bytes are stored in lower addresses. The integer value 258 is stored in memory as: 00000001 00000010.

150. Answer: 556
Explanation:
The integer value 300 in binary notation is: 00000001 00101100. It is stored in memory (small-endian) as: 00101100 00000001. Result of the expression *++ptr = 2 makes the memory representation as: 00101100 00000010. So the integer corresponding to it is 00000010 00101100 => 556.

151. Answer: 0
Explanation:
After 'ptr' reaches the end of the string the value pointed by 'str' is '\0'. So the value of 'str' is less than that of 'least'. So the value of 'least' finally is 0.

152. Answer: (char*(*)( )) (*ptr[N])( );

153. Answer: Compiler Error: Undefined structure date
Explanation:
Inside the struct definition of 'student' the member of type struct date is given. The compiler doesn't have the definition of date structure (forward reference is not allowed in C in this case) so it issues an error.

154. Answer: Compiler Error: Undefined structure date
Explanation:
Only declaration of struct date is available inside the structure definition of 'student' but to have a variable of type struct date the definition of the structure is required.

155. Explanation:
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.

156. Answer: No
Explanation:
Functions can only pass pointers and not arrays. The numbers that are allowed inside the [] is just for more readability. So there is no difference between the two declarations.

157. Answer & Explanation:
If the body of the loop never executes p is assigned no address. So p remains NULL where *p =0 may result in problem (may rise to runtime error "NULL pointer assignment" and terminate the program).

158. Answer & Explanation:
assert macro should be used for debugging and finding out bugs. The check s != NULL is for error/exception handling and for that assert shouldn't be used. A plain if and the corresponding remedy statement has to be given.

159. Answer & Explanation:
Assert macro is used for debugging and removed in release version. In assert, the experssion involves side-effects. So the behavior of the code becomes different in case of debug version and the release version thus leading to a subtle bug.
Rule to Remember:
Don't use expressions that have side-effects in assert statements.

160. Answer: Undefined behavior
Explanation:
The second statement results in undefined behavior because it points to some location whose value may not be available for modification. This type of pointer in which the non-availability of the implementation of the referenced location is known as 'incomplete type'.

161. Answer: No output
Explanation:
The else part in which the printf is there becomes the else for if in the assert macro. Hence nothing is printed.
The solution is to use conditional operator instead of if statement,

```
#define assert(cond) ((cond)?(0): (fprintf (stderr, "assertion
        failed: \ %s, file %s, line %d \n",#cond,
        __FILE__,__LINE__), abort()))
```

Note:
However this problem of "matching with nearest else" cannot be solved by the usual method of placing the if statement inside a block like this,

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

162. Answer: No
Explanation:
Is it not legal for a structure to contain a member that is of the same
type as in this case. Because this will cause the structure declaration to be recursive without end.

163. Answer: Yes.
Explanation:
*b is a pointer to type struct a and so is legal. The compiler knows, the size of the pointer to a structure even before the size of the structure
is determined(as you know the pointer to any type is of same size). This type of structures is known as 'self-referencing' structure.

164. Answer: No
Explanation:
The typename aType is not known at the point of declaring the structure (forward references are not made for typedefs).

165. Answer: Yes
Explanation:
The typename aType is known at the point of declaring the structure, because it is already typedefined.

166. Answer: No
Explanation:
When the declaration,
typedef struct a aType;
is encountered body of struct a is not known. This is known as 'incomplete types'.

167. Answer :
sizeof (void *) = 2
sizeof (int *) = 2
sizeof (double *) = 2
sizeof(struct unknown *) = 2
Explanation:
The pointer to any type is of same size.

168. Answer & Explanation:
The second one is better because gets(inputString) doesn't know the size of the string passed and so, if a very big input (here, more than 100 chars) the charactes will be written past the input string. When fgets is used with stdin performs the same operation as gets but is safe.

169. Answer & Explanation:
Prefer the first one. If the str contains any format characters like %d then it will result in a subtle bug.

170. Answer: Compiler Error: "Unexpected end of file in comment started in line 5".
Explanation:
The programmer intended to divide two integers, but by the "maximum munch" rule, the compiler treats the operator sequence / and * as /* which happens to be the starting of comment. To force what is intended by the programmer,

```
int k = *ip/ *jp;
// give space explicity separating / and *
//or
int k = *ip/(*jp);
// put braces to force the intention
```

will solve the problem.

171. Answer: Implementaion dependent
Explanation:
The char type may be signed or unsigned by default. If it is signed then ch++ is executed after ch reaches 127 and rotates back to -128. Thus ch is always smaller than 127.

172. Answer: Yes
Explanation:
The pointer ptr will point at the integer in the memory location 0x400.

173. Answer: Compiler error: Too many initializers
Explanation:
The array a is of size 4 but the string constant requires 6 bytes to get stored.

174. Answer: HELL%@!~@!@???@~~!
Explanation:
The character array has the memory just enough to hold the string "HELL" and doesnt have enough space to store the terminating null character. So it prints the HELL correctly and continues to print garbage values till it accidentally comes across a NULL character.

175 Answer: Compiler error: Cannot increment a void pointer
Explanation:
Void pointers are generic pointers and they can be used only when the type is not known and as an intermediate address storage type. No pointer arithmetic can be done on it and you cannot apply indirection operator (*) on void pointers.

176. Answer:

177. Answer: Variable length argument lists

178. Answer: string constant
Explanation:
The program suffers no problem and gives the output correctly because the character constants are stored in code/data area and not allocated in stack, so this doesn't lead to dangling pointers.

179. Answer: Garbage values.

Explanation:

Both the functions suffer from the problem of dangling pointers. In someFun1() temp is a character array and so the space for it is allocated in heap and is initialized with character string "string". This is created dynamically as the function is called, so is also deleted dynamically on exiting the function so the string data is not available in the calling function main() leading to print some garbage values. The function someFun2() also suffers from the same problem but the problem can be easily identified in this case.

180. Answer: Behavior is implementation dependent.

Explanation:

The detail if the char is signed/unsigned by default is implementation dependent. If the implementation treats the char to be signed by default the program will print –128 and terminate. On the other hand if it considers char to be unsigned by default, it goes to infinite loop.

Rule:

You can write programs that have implementation dependent behavior. But dont write programs that depend on such behavior.

181. Answer

Definition.
x is a pointer to array of(size 10) integers.

Apply clock-wise rule to find the meaning of this definition.

182. Ans. (c)
183. Ans. (a)
184. Ans. (b)
185. Ans. (d)
186. Ans. (c)
187. Ans. (a)
188. Ans. (a)
189. Ans. (iii)
190. Ans. (c)