1. ```
   typedef struct{
       char *;
       nodeptr next;
       } * nodeptr;
       what does nodeptr stand for?
       ans:
   ```

2. `int *x[](); means`

   **ans:expl: Elments of an array can't be functions.**

3. ```
   struct list{
       int x;
       struct list *next;
       }*head;
       the struct head.x =100
       Ans: above is correct / wrong
   ```
   expl: **Before using the ptr type struct variable we have to give memory
   to that .And also when ever the struct variable is ptr then we access the
   members by "->" operator.**

4. 
   ```
   main()
       {
       int i;
       i=1;
       i=i+2*i++;
       printf(%d,i);}
   ```
   **ans: 4**

5. 
   ```
   main()
       {
            FILE *fp1,*fp2;
           fp1=fopen("one","w")
           fp2=fopen("one","w")
       fputc('A',fp1)
           fputc('B',fp2)
           fclose(fp1)
           fclose(fp2)}
           a.error b. c. d.
   ```
   **ans: no error. But It will over writes on same
file.**

6. 
   ```
   #include<malloc.h>
               char *f()
       {char *s=malloc(8);
               strcpy(s,"goodbye");}
        main()
   ```

```
{
 char *f();
 printf("%c",*f()='A');}
 o/p=?
```

**for strcpy function string.h header file should be included**
**semicolon is missing in strcpy function**
**leftside function call can come when it is returning some pointer so          *p='A';**


7.   ```
     #define MAN(x,y) (x)>(y)?(x):(y)
     main()
          { int i=10,j=5,k=0;
          k= MAX(i++,++j);
          printf("%d %d %d ",i,j,k);
          }
     ```

     **ans. 12 6 11**

8.   ```
     main()
          {
          int a=10,b=5, c=3,d=3;
          if (a<b)&&(c=d++)
          printf("%d %d %d %d", a,b,c,d);
          else
          printf("%d %d %d %d", a,b,c,d);
          }
     ```

     **ans: 10 5 3 3   Note: if condition should be in braces**

9.   ```
     main()
          {
          int i = 10;
          printf(" %d %d %d \n", ++i, i++, ++i);
          }
     ```

     **ans: 13 11 11**

10.  ```
     main()
          {
               int *p, *c, i;
               i = 5;
               p = (int*) (malloc(sizeof(i)));
               printf("\n%d",*p);
          *p = 10;
          printf("\n%d %d",i,*p);
          c = (int*) calloc(2);
     ```

```
            printf("\n%d\n",*c);
            }
```

**Note: calloc function has less parameters   calloc(n, elemsize)**

```
      main()
            {
            int *p, *c, i;
            i = 5;
            p = (int*) (malloc(sizeof(i)));
            printf("\n%d",*p);
            *p = 10;
            printf("\n%d %d",i,*p);
            c = (int*) calloc(2,2);
            printf("\n%d\n",*c);
            }
```
**ans: garbage, 5, 10, 0 (malloc gives garbage and calloc initializes with zeros)**

11.   #define MAX(x,y) (x) >(y)?(x):(y)
```
             main()
            {
            int i=10,j=5,k=0;
                 k= MAX(i++,++j);
                 printf("%d..%d..%d",i,j,k);
                 }
```

            **ans: 12 6 11**

12.   main()
```
                {
                enum  _tag{ left=10, right, front=100, back};
            printf("left is %d, right is %d, front is %d, back is
%d",left,right,front,back);
                }
```

                **ans: left is 10, right is 11, front is 100, back is 101**

13.   main()
```
            {
             int a=10,b=20;
             a>=5?b=100:b=200;
             printf("%d\n",b);
            }
```

            **ans: lvalue required for ternary operator**

14.   #define PRINT(int) printf("int = %d ",int)

```
        main()
            {
            int x,y,z;
            x=03;y=02;z=01;
            PRINT(x^x);
            z<<=3;PRINT(x);
            y>>=3;PRINT(y);
            }
```

**ans: int = 0 int = 3 int = 0**

15.  main()
```
            {
            char s[] = "Bouquets and Brickbats";
            printf("\n%c, ",*(&s[2]));
            printf("\n%s, ",s+5);
            printf("\n%s,",s);
            printf("\n%c",*(s+2));
            }
```

**ans: u,**
   **ets and Brickbats,**
   **Bouquets and Brickbats,**
   **u**

16.  main()
```
            {
            struct s1
            {
            char *str;
            struct s1 *ptr;
            };
            static struct s1 arr[] = { {"Hyderabad",arr+1},
                {"Bangalore",arr+2},
                {"Delhi",arr}
                };
                struct s1 *p[3];
                int i;
                for(i=0;i<=2;i++)
                p[i] = arr[i].ptr;

                printf("%s\n",(*p)->str);
                printf("%s\n",(++*p)->str);
                printf("%s\n",((*p)++)->str);

            }
```

**ans: Bangalore**
   **Delhi**
   **Delhi**

17.  main()

```
{
char *p = "hello world!";
p[0] = 'H';
printf("%s",p);
}
```

**ans: Hello world**

18.  ```
     main()
     {
      int x=1,y=1;
      while( (x > 0) && (y > 0) )
      {
       printf("%16d%16d",x,y);
       x += y;
       y += x;
       }
     }
     ```

**ans: here x = x+y and y = x+2y when y goes beyond 32767
it falls in —ve side and loop breaks**

19.  ```
     int f(int p)
     {
     int i = 0, t = 1, s = 1;
     while( s <=
p)
         {
         i++;
         t += 2;
         s += s;
         }
     return i;
     }
     ```

**ans: this function gives the no. of bits required to represent a number in binary form**

20. remove the duplicate from a sorted array.

21. fibonacci series upto 100 recursively.

22. main()

```
{
char c[]={ " enter" , "first" , "print" , "new" }.;
char **cp[]={c+3, c+2, c+1, c};
char ***cpp[]=cp;
printf("%s", ++*cp);
printf("%s",--*++cp);
}
```

    **ans: lvalue required for second printf statement**

23. GCD and LCM programs

24. Write a program to print
    1
    2 2
    3 3 3
    4 4 4 4
    5 5 5 5 5.

    **ans:**
    **main()**
    ```
    {
    int i,j;
    for(i=1;i<=5;i++)
    {
    printf("\n");
    for(j=i;j>0;j--)
    printf("%d",i);
    }
    ```

25. double what( double z, int y)
    ```
    {
    double answer = 1;

    while( y > 0 )
    {
    ```

```
        if( y%2 == 1)
        answer = answer * z;
        y=y/2;
        z=z*z;
        }
        return answer;
        }
```

**ans: z power y**


26.    Program for square root.


27.    Write program to print
       1
       2 3
       4 5 6
       7 8 9 10
       11 12 13 14

**ans:**

**main()**
```
        {
        int i,j,k;
             k = 1;
        for(i=1;i<=5;i++)
        {
        for(j=i;j>0;j--)
        printf("%d",k++);
             printf("\n");
        }
```

28.    write a function maxsubstring(str,alpha,theta) str is the
       source string and have to return maximum substring which
       starts with alpha and ends with theta.

**ans:**

**main()**
```
        {
        int i,j=0,k;
        char st = 'x';
        char en = 'y';
        char p[]="abxabcdyxabcdabcydabcdxabycd";
        char *str;

        for(i=0;p[i]!='\0';i++)
        {
```

```
          if(p[i] == st)
          break;
          }
          if(p[i]=='\0')
          {
          printf("\n starting character not found\n");
          exit(0);
          }
          str = &p[i];
          k=i;
          while(p[++i]!='\0')
          if(p[i] == en)
          j=i;
          if(j==0)
          printf(" ending character not found\n");
          else
          for(;k<=j;k++)
          printf("%c",*str++);
          }
```

29.   How do you write a program which produces its own source code
as its output?

        How can I find the day of the week given the date?

        Why doesn't C have nested functions?

        What is the most efficient way to count the number of bits
which are set in a value?

        **ans: K. Ritchie**

        How can I convert integers to binary or hexadecimal?

        **ans: K. Ritchie**

        How can I call a function, given its name as a string?

        **ans: function pointers**

        How do I access command-line arguments?

        How can I return multiple values from a function?

        **ans: using pointer or structures**

        How can I invoke another program from within a C program?

**ans: using system function**

How can I access memory located at a certain address?

How can I allocate arrays or structures bigger than 64K?

How can I find out how much memory is available?

How can I read a directory in a C program?

How can I increase the allowable number of simultaneously open files?

What's wrong with the call "fopen("c:\newdir\file.dat", "r")"?

30.  ```c
     main()
          {
          int x=10,y=15;
          x=x++;
          y=++y;
          printf("%d %d\n",x,y);
          }
     ```

     **ans: 11 16**

31.  ```c
     int x;
     main()
          {

          int x=0;
          {
          int x=10;
          x++;
          change_value(x);
          x++;
          Modify_value();
          printf("First output: %d\n",x);
          }
          x++;
          change_value(x);
          printf("Second Output : %d\n",x);
          Modify_value();
          printf("Third Output : %d\n",x);
          }

     Modify_value()
     {
     return (x+=10);
     }
     ```

```
change_value()
{
return(x+=1);
}
```

**ans:**

**First output : 12**
**Second output : 1**
**Third output : 1**

32.  `main()`
```
{
int x=20,y=35;
x = y++ + x++;
y = ++y + ++x;
printf("%d %d\n",x,y);
}
```

**ans: 57 94**

33.  `main()`
```
{
char *p1="Name";
char *p2;
p2=(char *)malloc(20);
while(*p2++=*p1++);
printf("%s\n",p2);
}
```

**ans: No output since p2 is at null character to get**
**output modify the program given below. (Note: <malloc.h>**
**should be included)**

```
{
char *p1="Name";
char *p2,*p3;
p2=(char *)malloc(20);
p3=p2;
while(*p2++=*p1++);
printf("%s\n",p3);
}
```

34.  `main()`
```
{
int x=5;
printf("%d %d %d\n",x,x<<2,x>>2);
```

```
        }
```

**ans: 5 20 1**


35.  `#define swap1(a,b) a=a+b;b=a-b;a=a-b;`
     `main()`
```
        {
        int x=5,y=10;
        swap1(x,y);
        printf("%d %d\n",x,y);
        swap2(x,y);
        printf("%d %d\n",x,y);
        }
```
     `int swap2(int a,int b)`
```
        {
        int temp;
        temp=a;
        b=a;
        a=temp;
        return;
        }
```

**ans:**
**10 5**
**10 5**


36.  `main()`
```
        { char *ptr = "Ramco Systems";
        (*ptr)++;
        printf("%s\n",ptr);
        ptr++;
        printf("%s\n",ptr);
        }
```

**ans:**
**Samco Systems**
**amco Systems**


37.  `main()`
```
        { char s1[]="Ramco";
        char s2[]="Systems";
        s1=s2;
        printf("%s",s1);
        }
```

**ans: lvalue required (s1 is base address of array)**

38.  main()
```
{
char *p1;
char *p2;
p1=(char *) malloc(25);
p2=(char *) malloc(25);
strcpy(p1,"Ramco");
strcpy(p2,"Systems");
strcat(p1,p2);
printf("%s",p1);
}
```

**ans: RamcoSystems  (Note: <malloc.h>  should be included)**

39.  A code like this is given.
a. for(i=0;i<num;i++)
b. for(i=num;i>0;i--)
Assuming no code optimization and assume that the microprocessor
has flags etc. which one is faster.

**Ans: b will execute faster.**

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

**ans: 1, 2**

41.  main()
```
{
struct
{
char a[3];
int b;
}x;
char *cp;
printf("%d %d",sizeof(cp),sizeof(x));
}
```

**ans: 4 5    since pointer cp stores address(32-bit) 4 bytes it takes and and x takes 5 bytes(3 for character array a and 2 for int b)**

42. main()
```
        {
        int p=3,q=4;
        q = shw(&p);
        printf("%d %d",p,q);
        }

        int shw(int *a)
        {
         *a = 10;
        }
```

**ans: 10 garbage**


43. write 7*a interms of +,-,<<

**ans: (x<<3-x)**

44. main()
```
        {
        char *s1 = "hello",*s2 ="abce";
        strcpy(s1,"");
        s2[0] = s1[0];
        printf("%d%d",strlen(s1),strlen(s2));
        }
```

**ans: 0 0**


45. main()
```
        {
        int i=10;
        printf("%d%d%d",i,i++,++i);
        }
```

**ans: 12 11 11 (compiler dependent)**



46. const char *
    char * const
    What is the differnce between the above two?

**ans: const char *        pointer to a constant character**
**        char * const        constant pointer pointing to a**
**character**

47. main()
```
        {
```

```
char *x="new";
char *y="dictonary";
char *t;
void swap (char * , char *);
swap (x,y);
printf("(%s, %s)",x,y);

char *t;
t=x;
x=y;
y=t;
printf("-(%s, %s)",x,y);
}
void swap (char *x,char *y)
{
char *t;
y=x;
x=y;
y=t;
}
```

**ans: multiple declaration of t and all declarations should be before executable statement(errors)**


48.  main()
```
{
char p[]="string";
char t;
int i,j;
for(i=0,j=strlen(p);i<j;i++)
{
t=p[i];
p[i]=p[j-i];
p[j-i]=t;
}
printf("%s",p);
}
```

**ans: will not print anything since p will be pointing to a null string**


49.  main()
```
{
int i=10;
printf("%d %d %d",i,++i,i++);
}
```

**ans: 12 12 10 (compiler dependent)**

50. main()

```
{
void f(int,int);
int i=10;
f(i,i++);
}
void f(int i,int j)
{
if(i>50)
return;
i+=j;
f(i,j);
printf("%d,",i);
}
```

**ans: 51 41 31 21 (i=11, j=10 for function 'f')**

51. main()

```
{
void f(int,int);
int i=10;
f(i,++i);
}
void f(int i,int j)
{
if(i>50)
return;
i+=j;
f(i,j);
printf("%d,",i);
}
```

**ans: 55 44 33 22 (i=11, j=11 for function 'f')**

52. main()

```
{
char *s="hello world";
int i=7;
printf("%.*s",i,s);
}
```

**ans: hello w**

53. main()

```
{
int a,b;
printf("enter two numbers :");
scanf("%d%d",a,b);
printf("%d+%d=%d",a,b,a+b);
}
```

**ans: will generate run time error /core dump**

54.  main()
```
       {
       union{
       int x;
       char y;
       struct {
       char x;
       char y;
       int xy;}p;
       }q;
       printf("\n %d,%d",sizeof(q),sizeof(int));
       }
```

**ans: 4,2**

55.  main()
```
       {
       char *x="String";
       char y[] = "add";
       char *z;
       z=(char *) malloc(sizeof(x)+sizeof(y)=1);
       strcpy(z,y);
       strcat(z,x);
       printf("%s+%s=%s",y,x,z);
       }
```

**ans: add+String=addString**

56.  an arrey of n pointers to function returning pointers to functions returning pointers to characters

**ans: char * (* (*x[n]) ()  ) ()**

pointer to array of int, char etc.,          this is array pointer

**ans: int (*x)[]      char (*x)[]**

array of pointer to int, char etc.,          this is pointer array

**ans: int *x[]          char *x[]**

function returning pointer to int, char etc.,

**ans: int *x()      char *x()**

pointer to function returning int, char etc.,

**ans: int (\*x)()    char (\*x)()**

function returning pointer to array of pointer to function returning char

**ans: char (\*(\*x()) []) ()**

array of pointer to function returning pointer to array of char

**ans: char (\*(\*x[]) () ) []**

57.  main()
```
     {
     enum number { a=-1, b= 4,c,d,e};
     printf("%d",e);
     }
```

**ans: 7**

58.  main()
```
     {
     int i=0;
     for(i=0;i<20;i++)
     {
     switch(i)
     {2
     case 0:i+=5;
     case 1:i+=2;
     case 5:i+=5;
     default: i+=4;
     break;}
     printf("%d,",i);
     }
     }
```

**ans: 16,21 (after case and default colon should be there)**

59.  main()
```
     {
     int i, count, x=1;
     for(i=0, count=0;i<16;i++)
     if( !(x&(1<<i)) )
     count++;
     printf("%d",count);
     }
```

**ans: 15 (no. of zeros)**

```
60.  main()
        {
        int i, count, x=1;
        for(i=0, count=0;i<16;i++)
        if(x&(1<<i) )
        count++;
        printf("%d",count);
        }

        ans: 1 (no. of ones)
```

61.  which one will over flow given two programs
     prog 1: prog2:

```
main() main()
{ {
int fact; int fact=0
long int x; for(i=1;i<=n;i++)
fact=factoral(x); fact=fact*i;
} }

int factorial(long int x)
{
if(x>1) return(x*factorial(x-1);
}
```

     **ans: program 1 (program 2 is always zero since fact =0)**

```
62.  main()
        {
        char str[5]="hello";
        if(str==NULL) printf("string null");
        else printf("string not null");
        }

        ans: string not null
```

```
63.  void f(int value)
        {
        for (i=0;i<16;i++)
        {
        if(value &0x8000>>1) printf("1")
        else printf("0");
        }
        }

        ans: binary output of value
```

```
64.  void f(int *p)
     {
```

```
static val=100;
val=&p;
}
main()
{
int a=10;
printf("%d ",a);
f(&a);
printf("%d ",a);
}
```

**ans: nonportable pointer conversion (we can't store address in integer variable, we have to take pointer to store address)**

65. `main()`
```
{
int x, *y;
x = y;
printf("%d",x);
}
```

**ans: nonportable pointer conversion**

66. `# define f(a,b) a+b`
```
#define g(c,d) c*d
find value of f(4,g(5,6))
```

**ans: 34**

67. `main()`
```
{
char a[10]="hello";
strcpy(a,'\0');
printf("%s",a);
}
```

**ans: arguments must be a string constant or character array variable**
**here it is constat character not a string constant.**
**Hence program error**

68. `char a[5][15];`
```
int b[5][15];
address of a 0x1000 and b is 0x2000 find address of a[3][4]
and b[3][4]
interger takes 32-bits and character takes 8-bits
```

**ans: a[3][4] = 0x1031    b[3][4] = 0x20C4**
**(Note: addresses are in hexadecimal)**

69. Given an interger in binary form,find the number of ones in
    that number without counting each bit.(This questin is not
    multiple choice question. This question carries more
    marks. So please take care for this question.)

        **ans: K.Ritchie**

70. main()
```
    {
    a=2;
    b=3;
    x=SUM(a,b)*2;
    printf("x=%d\n",x);
    }
```

        **ans: 8**

71. number(int i)
```
    {
    number++;
    printf("%d\n",number);
    }

    main()
    {
    static int i=0;
    number(i);
    }
```

    **ans: lvalue required (function name is an address. So ++
    operator should not be applied)**

72. main()
```
        {
        unsigned char i;
        int sum;
        for(i=0; i<300; i++)
        sum+ = i;
        printf("\nSum = %d\n", sum);
        }
```

        **ans: infinite loop**

73. void fn(int *p)
```
    {
    static int val = 100;
    p = &val;
    }
```

```
main()
{
int i=10;
printf("i=%d\n", i);
fn(&i);
printf("i=%d\n", i);
}
```

**ans: i=10**
**i=10**

74.  Swapping without using a temporary variables. (3 methods)

**ans:**

```
x = x+y;
y = x-y;
x = x-y;

x = x^y;
y = x^y;
x = x^y;

x = x*y;
y = x/y;
x = x/y;
```

75.  Code 1 :
```
for(i=0; i<1000; i++)
for(j=0; j<100; j++)
x = y;
```

Code 2 :
```
for(i=0; i<100; i++)
for(j=0; j<1000; j++)
x = y;
```

Which code will execute faster

**ans: Code2      (Code 1 = 1,01000 increment operations)**
**(Code 2 = 1,00100 increment operations)**

76.  main()
```
{
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, i, x=10,
temp;
for(i=0; i<x; i++){
temp = a[i];
a[i] = a[x-i-1];
a[x-i-1] = temp;
}
```

**ans: remains same**

77.  ```
     main(0
         {
         int i = 1;
         fork();
         fork();
         printf("\ni = %d\n", i+1);
         }
     ```

     **ans: 4 printfs will occur and i = 2**

78.  ```
     #define MAX(a, b) a>b ? a:b
     main()
     {
     int m, n;
     m = 3 + MAX(2, 3);
     n = 2 * MAX(3, 2);
     printf("m = %d, n = %d\n", m, n);
     }
     ```

     **ans: m = 2, n = 3**

79.  ```
     main()
         {
         int i=10;
         fork();
         fork();
         fork();
         printf("%d",i);
         }
     ```

     **ans: 8 printfs will occur and i = 10 (2 power no. of
     forks times printfs)**

80.  ```
     #define f(a,b) a+b
     #define g(a,b) a*b

     main()
     {
     int m;
     m=2*f(3,g(4,5));
     printf("\n m is %d",m);
     }
     ```

     **ans: m is 26**

81.  ```
     main()
         {
     ```

```
        char a[10];
        strcpy(a,"\0");
        if (a==NULL)
        printf("\a is null");
        else
        printf("\n a is not null");
        }
```

**ans: a is not null**

82.  main()
```
        {
        char a[5]="hello";
        printf("%s",a);
        }
```

**ans: array size is small it should be 6**

83.  main()
```
        {
        unsigned int x=-1;
        int y;
        y = ~0;
        if(x == y)
        printf("same");
        else
        printf("not same");
        }
```

**ans: same (-1 is stored in 2's complement form)**

84.  char *gxxx()
```
     {
     static char xxx[1024];
     return xxx;
     }

     main()
     {
     char *g="string";
     strcpy(gxxx(),g);
     g = gxxx();
     strcpy(g,"oldstring");
     printf("The string is : %s",gxxx());
     }
```

**ans: The string is : oldstring**

85.  void myalloc(char *x, int n)
```
     {
     x= (char *)malloc(n*sizeof(char));
```

```
memset(x,'\0',n*sizeof(char));
}
main()
{
char *g="String";
myalloc(g,20);
strcpy(g,"Oldstring");
printf("The string is %s",g);
}
```

**ans: The string is Oldstring**

86.   main()
```
{
char p[]="String";
int x=0;
if(p=="String")
{
printf("Pass 1");
if(p[sizeof(p)-2]=='g')
printf("Pass 2");
else
printf("Fail 2");
}
else
{
printf("Fail 1");
if(p[sizeof(p)-2]=='g')
printf("Pass 2");
else
printf("Fail 2");
}
```

**ans: Fail 1Pass 2**

87.   A code which had some declarations of some data items. There
      were a couple of normal data items (char, int..) and some
      pointers as well and a malloc call. You have to find the total
      memory taken up in the stack (Hint: Pointers and all are
      allocated in heap, not in stack, so don't count them).Also in
      most of these questions, they were specifying that the OS was
      32 bit.

88.   A structure was given and it contained normal data as well as
      some bit-wise data. You had to find the total size taken up by
      the structure

89.   Pointer to a function which returned an array of char pointers

      **ans: char *((*x)() ) []**

90.  Value of 2 particular variables in C(MAXINT and some other
     constant)


91.  What do you need to do to open more than 10 files
     simultaneously in Microsoft Operating System?

     **ans: change stdio.h/change CONFIG.SYS/compiler dependent**

92.  main()
```
        {
        int i=7;
        i = i++*i++;
        printf("%d\n",i);
        i=7;
        printf("%d %d\n",i ,i++*i++);
        i=2;
        printf("%d %d\n" ,i,   i++*++i*i++*i++);
        i=1;
        printf("%d %d %d\n", i,    i++*i++,     i++*i++*++i*i+
+);
        i=1;
        printf("%d %d %d\n" ,i,    i++*i++,    i++*i++*++i*i+
+*i++*++i);
        }
```

        **ans:     51
             9 56
             6 160
             7 30 32
             9 56 1120**

93.  main()
```
        {
        int d ;
        int i=10;
        d =sizeof(++i);
        printf("%d",d);
        }
```

        **ans: 2**

94.  char *f();
     main()
```
        {
        char*a,*f();
        a=(char*)malloc(20*sizeof(char));
        a=f();
        printf("%s",a);
        }
        char *f()
```

```
        {static char n[20];
        strcpy(n,"Hello World");
        return(n);
        }

        ans: Hello World
```

95.   char *f();
      main()
```
        {
        char*a,*f();
        a=(char*)malloc(20*sizeof(char));
        a=f();
        printf("%s",a);
        }
        char *f()
        {char n[20];
        strcpy(n,"Hello World");
        return(n);
        }
```

        **ans: unpredictable output. auto variable address should
        not be returned. It will lose its scope when it comes
        out of the block.**

96.   char *f()
      main()
```
        {
        char *a,*f();
        a=f();
        printf("%s",a);
        }
        char *f()
        {return("Hello World");}
```

        **ans: Hello World**

97.   what is the error
      main()
```
        {int j=10;
        switch(j)
        {case 20:
        pritnf("Less than 20");
        break;
        case 30:
        printf("Less than 30");
        break;
        default:
        printf("hello");
        }
```

**ans: printf not pritnf  and one brace } is missing**

98.   which is valid :
      (i)char arr[10];
      arr="hello";
      (ii) char arr[]="hello";

      **ans: second is correct. In first lvalue required.**

99.   main()
            {
            char *str;
            str=(char*)malloc(20*sizeof(char));
            strcpy(str,"test");
            strcat(str,'!');
            printf("%s",str);
            }

            **ans: strcpy function arguments should be either a
            character array variable or a string constant.  Instead
            of '!' give "!"**

100.  How many times main is get called
      main()
            {
            printf("Jumboree");
            main();
            }

            **ans: till stack overflow**

101.  main()
            {
            int i;
            if(i=0)
            printf(" Hell ");
            else
            printf("Heaven");
            }

            **ans: Heaven**

102.  main()
            {
            int i,j;
            for(i=0,j=0;i<5,j<25;i++,j++);
            printf("%d %d",i,j);
            }

**ans: 25 25  A pair of expressions separated by a comma is evaluated left to right, and the type and value of the result are the type and value of the right operand. Here we've to consider j<25 condition.**

103.  1)pointer to a function.
      2)pointer to structure.
      3)static variable and difference b/w(const char *p,char const *p,const char* const p).
      4)pass by value & reference.
      5)string library functions(syntax).
      6)Write a program to compare two strings without using the strcmp() function.
      7)Write a program to concatenate two strings.
      8)Write a program to interchange 2 variables without using the third one.
      9)Write programs for String Reversal & Palindrome check .
      10)Write a program to find the Factorial of a number.
      11)Write a program to generate the Fibinocci Series.
      12)searching and sorting alogorithms with complexities.

104.  Theory question about far pointers.

      **ans:  Far pointers are 4 bytes in size and local pointers are 2 bytes in size. important: i saw in a previous question paper of accenture which is in the chetana database, some lady wrote that size of an integer in C is 2 bytes and for C++ it is 4 bytes. This is absurd.The size of types is entirely dependent on the compiler used.for DOS Turbo C sizeof int is 2 and float is 4 bytes for windows borland C,C++ size of int is 4 bytes for linux gcc, size of int is 2 bytes. All these depends on the Operating system.Please keep this in mind.**

105.  main()
```
      {
      char str[]={"hell"};
      int i;
      for(i=0;i<5;i++)
      printf("%c%c%c%c\n",str[i],i[str],*(str+i),*(i+str));
      }
```

      **ans:  hhhh
             eeee
             llll
             llll**

106.  inline function does type checking and so it is better than a macro

107.  main()
```
      {
```

```
        int i = 10;
        int j = i >> 10;
        printf("%d",j);
        }

        ans: 0
```

108.  char *str = "Hello";
      char arr[] = "Hello";
      arr++; // ERROR..its like a pointer constant
      *(arr + 1) = 's';
      cout<<arr; // o/p: Hsllo

109.  struct Date
```
            {
              int yr;
              int day;
              int month;
            } date1,date2;

            date1.yr = 2004;
            date1.day = 4;
            date1.month = 12;
            now how will you initialize date2 (without member
            by member assignment)
            ans: date2 = date1;
```

110.  main()
```
        {
        extern int a;
        printf("%d",a);;
        }
        int a=20;

        ans: 20
```

111.  main()
```
        {
        int a[5]={2,3};
        printf("\n %d %d %d",a[2],a[3],a[4]);
        }
```

        ans: 0 0 0   if there are fewer initializers
uninitialized variables are zero

112.  main()
```
          {
           inti=-3,j=2,k=0,m;
           m=++i&&++j||++k;
           printf("\n %d %d %d %d",i,j,k,m);
          }
```

**ans: -2 3 0 1**

113. ```c
main()
{
int a,b;
a=sumdig(123);
b=sumdig(123);
printf("%d %d",a,b);
}
sumdig(int n)
{
static int s=0;
int d;
if(n!=0)
{
d=n%10;
n=(n-d)/10;
s=s+d;
sumdig(n);
}
else return(s);
}
```

**ans: 6 12**

114. ```c
#define CUBE(x)   (x*x*x)
main()
{
int a,b=3;
a=CUBE(b++);
printf("\n %d %d",a,b);
}
```

**ans: 27 6**

115. ```c
main()
{
const int x=get();
printf("%d",x);
}
get()
{
return(20);
}
```

**ans: 20   for auto variables initializers can be
function calls or some expressions. But for static
initializers should constants or constant expressions.**

116. A function has this prototype void f1(int **x), How will you call this function?
    {a) int **a;        (b) int a;        (c) int *a;        (d) int a=5;
    f1(a); f1(&a); f1(&a); f1(&&a);

        **ans: int *a; f1(&a);**

117. main()
        {
        int l=1;
        for(;;)
        {
        printf("%d",l++);
        if(l>10)
        break;
        }
        }

        **ans: 12345678910**

118. main()
        {
        char str[5]="fast";
        static char *ptr_to_array = str;
        printf("%s",ptr_to_array);
        }

        **ans: error. for auto variables initializers can be
        function calls or some expressions. But for static
        initializers should constants or constant expressions.**

119. main()
        {
        char str[5]="fast";
        char *ptr_to_array = str;
        printf("%s",ptr_to_array);
        }

        **ans: fast. for auto variables initializers can be
        function calls or some expressions. But for static
        initializers should constants or constant expressions.**

120. main()
        {
        int i=10;
        fn(i);
        printf("%d",i);
        }

```
fn(int i)
{
return ++i;
}
```

**ans: 10**

121. `main()`
```
{
int i,j;
i=10;
j=sizeof(++i);
printf("%d",i);
}
```

**ans: 10**

122. `main()`
```
{
struct emp
{
char name[20];
int age;
float sal;
};
struct emp e = {"tiger"};
printf("\n %d %f",e.age,e.sal);
}
```

**ans: 0 0.000000 If there are fewer initializers in the list than members of the structure the trailing members are initialized with zero. There may not be more initializers than members.**

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

**ans: –128**

124. `typedef enum grade{GOOD,BAD,WORST,}BAD;`
```
main()
{
BAD g1;
g1=1;
printf("%d",g1);
}
```

125. 
```
#define STYLE1 char
main()
     {
     typedef char STYLE2;
     STYLE1 x;
     STYLE2 y;
     x=255;
     y=255;
     printf("%d %d\n",x,y);
     }
```

     **ans: -1 -1**

126. 
```
#ifdef TRUE
int I=0;
#endif

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

     **ans: error since i is not declared**

127. 
```
main(0
     {
     char *pDestn,*pSource="I Love You Daddy";
     pDestn=(char *)malloc(strlen(pSource));
     strcpy(pDestn,pSource);
     printf("%s",pDestn);
     free(pDestn);
     }
```

     **ans: I Love You Daddy**

128. 
```
main()
     {
     char a[5][5],flag;
     a[0][0]='A';
     flag=((a==*a)&&(*a==a[0]));
     printf("%d\n",flag);
     }
```

     **ans: 1**

129. 
```
main()
     {
     int i=5,j=5,k;
```

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

**ans: lvalue required (++i++)**

130. main()
```
{
int b=10;
int *p=&b;
*p++;
printf("%d",*p);
}
```

**ans: unknown value (value at memory location next to the memory location of b**

131. main()
```
{
int i=0,j=50
while (i<j)
{
if(<some condtn>)
{
<body of the loop>
i++
}
elseif(<some condtn>)
{    <body of the loop>
j--
}
else(<some condtn>)
{<body of the loop>
j--
}
}
```
How many times the body of the loopis going to be executed?

**Ans: 50 times**

132. How can you include a library code written in C++ in a source code written in C?
(Options are there)

**ans. Some cross-linked platform(compiler) is required for this.**

133. main()
```
{
int a[20],i;
```

```
for(i=0;i<20;i++)
{
a[i]=i;
}
for(i=0;i<20;i++)
{
a[i]=a[20-i];
}
for(i=0;i<20;i++)
printf("%d",a[i]);
}
```

**ans: unknown value 19 18 17 16 15 14 13 12 11 10 11 12 13 14 15 16 17 18 19**

134. main()
```
{
int a[20],i;
for(i=0;i<20;i++)
{
a[i]=i;
}
for(i=0;i<20;i++)
{
a[i]=a[20-i];
}
for(i=0;i<20;i++)
printf("%d",a[i]);
}
```

**ans: 19 18 17 16 15 14 13 12 11 10 10 11 12 13 14 15 16 17 18 19**

135. void abc(int a[])
```
{
int k=0;int j=50;
while(k<j)
{
if(a[i]>a[j])
k++;
else
j--;
}
How many times the loop will occur?
```

**Ans: 50 times**

136. main()
```
{
int a[]={5,4,3,2,1};
int x,y;
```

```
int *p=&a[2];
*p++;
x=++*p;
y=*(p++);
printf("%d %d",x,y);
}
```

**ans: 3 3**

137. `int a;`
`scanf("%f",&a);   is there any error or warning ?`

**ans. no compile time error but run time error**

138. `main()`
```
{int *p,*q;
p=(int *)1000;
q=(int *)2000;
printf("%d",(q-p));
}
```

**ans: 500**

139. When a 'C' function call is made, the order in which
parameters passed to the function are pushed into the stack is

**ans: right to left**

140. `main()`
```
{
extern int a;
a=10;
printf("%d",a);
}
int a=20;
```

**ans: 10**

141. `sizeof ()` operator is used for

**ans:  data type & veriable**

142. `main()`
```
main()
{
int i = 2;
printf("%d %d %d %d ",i, i++,i--,i++);
}
```

**ans: 3 2 3 2**

143. main()
```
{
int i = 2;
printf("%old %old %old %old ",i, i++,i--,i++);
}
```

   **ans: 3ld 2ld 3ld 2ld**

144. Scope of a global variable which is declared as static?

   **ans: File**

145. main()
```
{
printf(" Hello \o is the world ");
}
```

   **ans: Hello o is the world**

146. What is
   int *p(char (*s)[])

   **ans: p is a function which is returning a pointer to
   integer
   which takes arguments as pointer to array of characters.**

147. How will u print TATA alone from TATA POWER using string copy
   and concate commands in C?

   **ans: implement strstr function**

148. main()
```
{
int n = 1;
switch(n)
case 1:printf("CASE !");
case(2):printf("default");
break;
}
```

   **ans: error (case outside of switch and misplaced break)
   all keywords in c should start with small letters**

149. #define min((a),(b)) ((a)<(b))?(a):(b)
   main()
```
{
int i=0,a[20],*ptr;
ptr=a;
while(min(ptr++,&a[9])<&a[8])
i=i+1;
```

```
            printf("i=%d\n",i);
            }

            ans: i=5
```

150. ~(~0<<8)?
          **ans: Last 8 digits are 1's rest are 0's.**

151. struct x
     {
     int I;
     char s;
     };
     union
     {
     struct x y;
     double j;
     }z;
     main()
```
            {
            printf("%d",sizeof (z));
            }

            ans: 8
```

152. main()
```
            {
            char a[]={'1','2','3',0,'1','2','3'};
            printf("%s",a);
            }

            ans: 123
```

153. main()
```
            {
            int a[]={'1','2','3',0,'1','2','3'};
            printf("%s",a);
            }

            ans: 1
```

154. main()
```
            {
            #define x 10
            {
            printf("%d",x);
            }
            }

            ans: 10
```

155. main()
```
{
#define x 10
{
printf("%d",++x);
}
}
```

   **ans: lvalue required**

156. main()
```
{
char a[]="ABCDEFGH";
printf("%d",sizeof(a));
}
```

   **ans: 9**

157. main()
```
{
int i=(int*)0x1000;
printf("%d",i);
}
```

   **ans: nonportable pointer conversion**

158. main(int I)
```
{
printf("%d",I);
}
```

   **ans: 1 (command line arguments)**

159. main()
```
{
printf(" %d",printf("helloworld"));
}
```

   **ans: helloworld 10**

160. main()
```
{
int a[2][2][6]
{{2,3,4,5,6,7}
{……………..}}
printf("%u%u%u%u",a,*a,**a,***a);
```

   assume base address is 567895

   **ans: 567895, 567895, 567895,2 (a, a[0], a[0][0], a[0][0]
   [0])**

```
161.  main()
          {
          int a[2][2]={{2},{3}};
          printf("%d ",a[0][0]);
          printf("%d ",a[0][1]);
          printf("%d ",a[1][0]);
          printf("%d ",a[1][1]);
          }
```

**ans: 2 0 3 0**

```
162.  char strbuf[]="hello ";
      char *strptr="world ";
      strbuf="world ";
      strptr="hello";
```

**ans: error (use strcpy function)**

```
163.  char str1[]="hello";
      char str2[]="hello";
      the conditional string test (str1==str2)
      returns FALSE
```

**ans: use strcmp function**

```
164.  main()
          {
          int i;
          char *str4="123four";
          i=atoi(str4);
          printf("%d",i);
          }
```

**ans: 123**

```
165.  main()
          {
          char loop;
          for(loop='A';loop<='z';loop++)
          printf("%c",loop);
          }
```

**ans: print characters of ascii value from 65 to 112**

```
166.  main()
          {
          char s[]={'1','2','3',0,'1','2','3'};
          printf("%s",s);
```

```
        }
```

**ans: 123**

167. main()
```
        {
        char *p="Caritor";
        *++p;
        printf("%s",p);
        *++p;
        printf("%s",*p);
        }
```

**ans: aritor ritor**

168. How to print "%" symbol in printf?

**ans: printf("\%");**

169. What is the max no of char in command line arguments?

**ans:**

170. arithmetic Operation can't be performed on void pointers.

171. main()
```
        {
        char str1[]="HELLO";
        char str2[]="HELLO";
        if(str1==str2)
        printf("EQUAL");
        else
        printf("NOT EQUAL");
        }
```

**ans: NOT EQUAL (use strcmp function for comparing strings)**

172. main()
```
        {
        int s=5;
        printf("%d",s,s<<2,s>>2);
        }
```

**ans: 5**

173. main()
```
        {
        int s=5;
        printf("%d %d %d",s,s<<2,s>>2);
```

```
        }

        ans: 5 20 1
```

174. main()
```
        {
        int a[2][2]={2,3};
        printf("%d %d %d %d",a[0][0],a[0][1],a[1][0],a[1][1]);
        }

        ans: 2 3 0 0
```

175. main()
```
        {
        int i=-3,j=2,k=0,m;
        m= ++j&&++i&&++k;
        printf("%d %d %d %d",i,j,k,m);
        }

        ans: -2 3 1 1
```

176. main()
```
        {
        const int i=7;
        printf("%d",++i);
        }

        ans: cannot modify a constant object
```

177. #define I 6
```
    main()
        {
        printf("%d",++I);
        }

        ans: lvalue required
```

178. main()
```
        {
        int a[2][3][4]={{1,2,3,4,5,6,7,8,9,1,1,2},
{2,3,4,7,6,7,8,9,0,0,0,0}};
        printf("%d %d %d %d",a,*a,**a,***a);
        }

        ans: 1002 1002 1002 1 (array begins at address 1002)
```

179. main()
```
        {
        printf("%c",7["sundaram"]);
        }
```

**ans: m (a[i], i[a], a[2], 2[a])**

180. ```
main()
        {
        printf("%c","sundaram"[7]);
        }
```

**ans: m (a[i], i[a], a[2], 2[a])**

181. ```
main(int argc , char * argv[])
        {
        int i,j=0;
        for(i=0;i<argc;i++)
        j=j+atoi(argv[i]);
        printf("%d",j);
        }
```

**ans: 6 (if command line arguments are myprog 1 2 3)**

182. ```
main()
        {
        printf("%d",-1>>4);
        }
```

**ans: –1 (–1 is stored in 2's complement form when it is shifted sign bit is extended)**

183. ```
struct x
{
int i;
char c;
};

union y{
struct x a;
double d;
};
main()
        {
        printf("%d",sizeof(union y));
        }
```

**ans: 8 (union y is a kunion variable type. Sizeof operator takes input either a variable or a data type)**

184. ```
struct x{
char c1;
char c2;
int i;
short int j;
};
```

```
struct y{
short int j;
char c1;
char c2;
int i;
};
main()
    {
    printf("%d %d",sizeof (struct x),sizeof (struct y));
    }
```

**ans: 6 6 (struct x and struct y are structure variable
types. Sizeof operator takes input either a variable or
a data type)**

185. main()
```
    {
    int k=2,j=3,p=0;
    p=(k,j,p);
    printf("%d\n",p);
    }
```

**ans: 0 (comma operator)**

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

**ans: prints -10 to -1**

187. main()
```
    {
    unsigned int i=-1;
    printf("%d %u\n",i,i);
    printf("%u\n",i*-1);
    }
```

**ans:  -1 65535
        1**

188. main()
```
    {
    int **i;
    int *j=0;
    i=&j;
    if (NULL != i&& NULL != *i)
    {
    printf("I am here");
```

```
        }
        }
```

**ans: does not print anything**

189.  main()
```
        {
        int *j=(int *)0x1000;
        printf("%p",j);
        }
```

**ans: 0000 : 1000**

190.  main()
```
        {
        int *j=0x1000;
        printf("%p",j);
        }
```

**ans: 0000:1000**

191.  main()
```
        {
        int *j=(int *)0x1000;   (or) int *j=0x1000;
        printf("%d",j);
        }
```

**ans: 4096**

192.  main(int x)
```
        {
        printf("%d",x);
        }
```

**ans: 1 (command line arguments)**

if the name of the executable file is abc and the
command line is
given as
abc xyz
what is the output

**ans: 2**

193.  main()
```
        {
        char a[]={'1','2','3',0,'1','2','3'};
        printf(a);
        }
```

**ans: 123**

```
194.  #define const const
      void main(int argc)
              {
              const int x=0;
              }

              ans: runs fine

195.  main()
              {
              int a[]={5,6};
              printf("%d",a[1.6]);
              }

              ans: 6

196.  struct x
              {
              int i=0; /*line A*/
              };
              main()
              {
              struct x y; /*line B*/
              }

              ans: error (i is initialized in struct body)

197.  struct {
      int len;
      char *str
      }*p;
      ++p -> len

              ans: increments len

198.  main()
              {
              char a[]="abcdefghijklmnopqrstuvwxyz";
              printf("%d",sizeof(a));
              }

              ans: 27 (sizeof operator includes null character also,
              whereas strlen function excludes null character)

199.  main()
              {
              char a[]="abcdefghijklmnopqrstuvwxyz";
              char *p=a;
              printf("%d ",strlen(p));
              p+=10;
```

```
          printf("%d",strlen(a));
          }

          ans: 26 26
```

200.  main()
```
          {
          printf("%d",printf(" hello world "));
          }
```

          **ans:  hello world 13 (including two spaces)**

201.  what is the output of the following code, assuming that the
      array
      begins at location 5364875?

      main()
```
          {
          int a[2][3][4]={
          {2,1,4,3,6,5,8,7,0,9,2,2},
          {1,2,3,4,5,6,7,8,9,0,1,2}
          };
          printf("%u %u %u %u",a,*a,**a,***a);
          }
```

          **ans: 5364875,5364875,5364875,2**

202.  main()
```
          {
          char a =0xAA ;
          int b ;
          b = (int) a ;
          b = b >> 4 ;
          printf("%x",b);
          }
```

          **ans: fffa**

203.  What is the size of the array declared as double * X[5] ?

      **ans. 5 * sizeof ( double * )**

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

          **ans: 100**

```
204. main()
        {
        int a;
        printf("%d",scanf("%d",&a));
        }

        ans: it will wait for a character from keyboard. If u
enter any number
        it will print 1.

205. main()
        {
        printf("as");
        printf("\bhi");
        printf("is\n");
        }

        ans: ahiis (\b is backspace. So s is erased)

206. main()
        {
        unsigned short a=-1;
        unsigned char b=a;
        printf("%d %d ",a,b);
        }

        ans: -1 255 (%d format specifier)

207. main()
        {
        unsigned short a=-1;
        unsigned char b=a;
        printf("%u%d ",a,b);
        }

        ans: 65535 255 (%u format specifier)

208. #define maxval 5
     main()
        {
        int i=1;
        if(i-maxval)
        {
        printf("inside");
        }
        else
        {
        printf("out");
        }
        }
```

209. ```
     #define a 3+3
     #define b 11-3
         main()
         {
         printf("%d",a*b);
         }
     ```

     ans: 33

210. ```
     main()
         {
         int *i;
         int s=(int *)malloc(10*sizeof(int));
         for (i=0;i<10;i++)
         {
         printf("%d",i*i);
         }
         }
     ```

     ans: error (Nonportable pointer conversion and illegal
use pointer i*i)

211. array's base address is 1000....array is a[5][4]..then wat is
     de
     correct address of a[4][3]...Each element takes 4 bytes

     ans:1076

212. int a[5,6]
     how much memory will be allocated

     ans: doubt(if comma operator is considered 12 bytes will
     be allocated)

213. ```
     #define d 10+10
     main()
         {
         printf("%d",d*d);
         }
     ```

     ans: 120

214. ```
     main()
         {
         int i,j=1;
         for(i=0;i<10;i++);
         {
         j=j+i;
     ```

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

**ans: 10 11**

215.  `static char *i;`
      `i=malloc(sizeof(char));`
      `find the error;`

   **ans: malloc returns void (type casting is required (char
   *) )**

216.  `main()`
```
        {
        int i=0xaa;
        char *p;
        p=(char *)i;
        p=p>>4;
        printf("%x",p);
        }
```

   **ans: illegal use of pointer p=p>>4**

217.  `main()`
```
        {
        enum{sunday=-1,monday,wednesday};
        printf("%d %d",sizeof(wednesday),wednesday);
        }
```

   **ans: 2 1**

218.  ->How do you write a program which produces its own source
      code as its output?
      ->How can I find the day of the week given the date?
      ->Why doesn't C have nested functions?
      ->What is the most efficient way to count the number of bits
      which are set in a value?
      ->How can I convert integers to binary or hexadecimal?
      ->How can I call a function, given its name as a string?
      ->How do I access command-line arguments?
      ->How can I return multiple values from a function?
      ->How can I invoke another program from within a C program?
      ->How can I access memory located at a certain address?
      ->How can I allocate arrays or structures bigger than 64K?
      ->How can I find out how much memory is available?
      ->How can I read a directory in a C program?
      ->How can I increase the allowable number of simultaneously
      open files?
      ->What's wrong with the call "fopen("c:\newdir\file.dat",
      "r")"?
```

```
219.  void main()
          {
          int d=5;
          printf("%f",d);
          }

          ans: undefined

220.  void main()
          {
          int i;
          for(i=1;i<4;i++)
          switch(i)
          {
          case 1: printf("%d",i);break;
          {
          case 2:printf("%d",i);break;
          case 3:printf("%d",i);break;
          }
          }
          switch(i) case 4:printf("%d",i);
          }

          ans: 1234

221.  void main()
          {
          int i;
          for(i=1;i<4;i++)
          switch(i)
          {
          case 1: printf("%d",i);break;
          {
          case 2:printf("%d",i);break;
          case 3:printf("%d",i);break;
          }
          switch(i) case 4:printf("%d",i);
          }
          }

          ans: 123

222.  void main()
          {
          char *s="\12345s\n";
          printf("%d",sizeof(s));
          }

          ans: 4 (pointer takes 4 bytes here)
```

```
223. void main()
        {
        unsigned i=1; /* unsigned char k= -1 => k=255; */
        signed j=-1;   /* unsigned or signed int k= -1 =>k=65535
     */
        if(i<j)
        printf("less");
        else
        if(i>j)
        printf("greater");
        else
        if(i==j)
        printf("equal");
        }
```

        **ans: less**

224.  How do you declare an array of N pointers to functions
      returning pointers to functions returning pointers to
      characters?

        **ans: char *(*(*a[N])())();**

```
        typedef char *pc;     /* pointer to char */
        typedef pc fpc();  /* function returning pointer to char
        */
        typedef fpc *pfpc;     /* pointer to above */
        typedef pfpc fpfpc();    /* function returning... */
        typedef fpfpc *pfpfpc;    /* pointer to... */
        pfpfpc a[N];          /* array of... */
```

225. int f();
     void main()
        {
        f(1);
        f(1,2);
        f(1,2,3);
        }
        f(int i,int j,int k)
        {
        printf("%d %d %d ",i,j,k);
        }

        **ans: 1 garbage garbage 1 2 garabage 1 2 3**

226. void main()
        {
        int count=10,*temp,sum=0;
        temp=&count;
        *temp=20;
        temp=&sum;
```

```
        *temp=count;
        printf("%d %d %d ",count,*temp,sum);
        }
```

**ans: 20 20 20**

227.  main()
```
        {
        static i=3;
        printf("%d",i--);
        return i>0 ? main():0;
        }
```

**ans: 321**

228.  char *foo()
```
        {
        char result[100];
        strcpy(result,"anything is good");
        return(result);
        }
        void main()
        {
        char *j;
        j=foo();
        printf("%s",j);
        }
```

**ans: anything is good (address of auto variable should
not be returned. Sometimes it will give unknown results)**

229.  void main()
```
        {
        char *s[]={ "dharma","hewlett-packard","siemens","ibm"};
        har **p;
        p=s;
        printf("%s ",++*p);
        printf("%s ",*p++);
        printf("%s ",++*p);
        }
```

**ans: harma harma ewlett-packard**

230.  main()
```
        {
        static int i = 0;
        int z;
        if(i++<5)
        {
        printf("%d ",i);
        }
```

```
else
exit(0);
z=3;
printf("%d %d ",z,main());
}
```

**ans: 1 2 3 4 5**

231. main()
```
{
static int i = 0;
int z;
if(i++>5)
{
printf("%d ",i);
exit(0);
}
z=3;
printf("%d %d ",z,main());
}
```

**ans: 7**

232. main()
```
{
int z=3;
printf("%d %d &",z,main());
}
```

**ans: infinite loop or till stack overflows**

233. main()
```
{
int i=3,j=5;
while (i--,j--)
{
printf("%d %d \n",i,j);
}
}
```

**ans:  2 4**
**     1 3**
**     0 2**
**     -1 1**
**     -2 0**
**     5 times loop will be executed**

234. main()
```
{
int i=3,j=5;
if(i--,j--)
```

```
        printf("%d %d \n",i,j);
        }

        ans: 2 4

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

        ans: 8 8 8

236.  main()
        {
        int times =5;
        int i=3;
        int j=4;
        int k=34;
        i=j+k;
        while(times --)
        {
        i=times;
        j=times;
        k=times;
        }
        printf("%d %d %d ",i,j,k);
        }

        ans: 0 0 0

237.  main()
        {
        int num =32765;
        while (num++);
        printf("%d ",num);
        }

        ans: 1

238.  main()
        {
        float k=3.4156;
        printf("%f %f ",floor(k),ceil(k));
        }

        ans: 3.000000 4.000000

239.  main()
        {
        int number =25;
```

```
            char name ='A';
            printf("The addition of the name and the number is %o
",name+number);
            }

            ans: The addition of the name and the number is 132
```

240.  The following function gives some error. What changes have to
be made

```
      void ( int a,int b)
            {
            int t; t=a; a=b; b=t;
            }
```

            ans: change everywhere a to *a and b to *b

241.  int main()

```
            {
            FILE *fp;
            fp=fopen("test.dat","w");
            fprintf(fp,'hello\n");
            fclose(fp);
            fp=fopen ("test.dat","w");
            fprintf (fp, "world");
            fclose(fp);
            return 0;
            }
```

      If text.dat file is already present after compiling and
      execution how many bytes does the file occupy ?

            ans:   5 bytes

242.  main()

```
            {
            int i;
            for(i=0;i<20;i++)
            {
            switch(i)
            {
            case 0:i+=5;
            case 1:i+=2;
            case 5:i+=5;
            default: i+=4;
            break;}
            printf("%d,",i);
            }
            }
```

            ans: 16, 21,

243.  main()

```
        {
        char c=-64;
        int i=-32;
        unsigned int u =-16;
        if(c>i)
        {
        printf("pass1,");
        if(c<u)
        printf("pass2");
        else
        printf("Fail2");
        }
        else
        printf("Fail1,");
        if(i<u)
        printf("pass2");
        else
        printf("Fail2");
        }

        ans: Fail1, pass2
```

244.  main()
```
        {
        char c=-64;
        int i=-32;
        unsigned int u =16;
        if(c>i)
        {
        printf("pass1,");
        if(c<u)
        printf("pass2");
        else
        printf("Fail2");
        }
        else
        printf("Fail1,");
        if(i<u)
        printf("pass2");
        else
        printf("Fail2");
        }

        ans: Fail1, Fail2 (check with above program)
```

245.  void main()
```
        {
        int i;
        char a[]="String";
        char *p="New Sring";
        char *Temp;
```

```
            Temp=a;
            a=malloc(strlen(p) + 1);
            strcpy(a,p); //Line number:9//
            p = malloc(strlen(Temp) + 1);
            strcpy(p,Temp);
            printf("(%s, %s)",a,p);
            free(p);
            free(a);
            } /*Line number 15*/
```

**ans: lvalue required (at line no. 8)**

246. main()
```
            {
            unsigned int x=-1;
            int y;
            y = ~0;
            if(x == y)
            printf("same");
            else
            printf("not same");
            }
```

**ans: same**

247. char *gxxx()
```
            {
            static char xxx[1024];
            return xxx;
            }

            main()
            {
            char *g="string";
            strcpy(gxxx(),g);
            g = gxxx();
            strcpy(g,"oldstring");
            printf("The string is : %s",gxxx());
            }
```

**ans: The string is oldstring**

248. void myalloc(char *x, int n)
```
            {
            x= (char *)malloc(n*sizeof(char));
            memset(x,'\0',n*sizeof(char));
            }

            main()
            {
            char *g="String";
```

```
        myalloc(g,20);
        strcpy(g,"Oldstring");
        printf("The string is %s",g);
        }
```

**ans: The string is Oldstring**

249.  main()
```
        {
        char p[]="String";
        int x=0;
        if(p=="String")
        {printf("Pass 1");
        if(p[sizeof(p)-2]=='g')
        printf("Pass 2");
        else
        printf("Fail 2");
        }
        else
        {
        printf("Fail 1");
        if(p[sizeof(p)-2]=='g')
        printf("Pass 2");
        else
        printf("Fail 2");
        }
        }
```

**ans: Fail 1Pass 2 (address of array and address of
string where it is stored are different)**

250.  main()
```
        {
        char *p="String";
        int x=0;
        if(p=="String")
        {printf("Pass 1");
        if(p[sizeof(p)-2]=='g')
        printf("Pass 2");
        else
        printf("Fail 2");
        }
        else
        {
        printf("Fail 1");
        if(p[sizeof(p)-2]=='g')
        printf("Pass 2");
        else
        printf("Fail 2");
        }
        }
```

251. main()
```
{
printf("%u",main);
}
```

ans: 0

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

ans: starting address of main function x:y (segment :
offset). Each time u run starting address will change.
Function name always gives starting address of that
function.

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

ans: infinite loop or till stack overflows. main
function is called recursively infinite times or till
stack overflows

253. main()
```
{
int i=10;
printf("%d %d %d",i,i++,++i);
}
```

ans: 12 11 11 (compiler dependent)

254. main()
```
{
int *p,*q;
p=(int *)1000;
q=(int *)2000;
printf("%d",(q-p));
}
```

ans: 500

255. find(int x,int y)
```
{return ((x<y)?0:(x-y)):}
find(a,find(a,b)) is used for?
```

**ans: find out minimum of a, b**

256. `find(int x,int y);`

```
main()
      {
      int x,a=8,b=6;
      x=find(a,find(a,b));
      printf("%d",x);
      }

find(int x,int y)
{ return ((x<y)?0:(x-y));}
```

**ans: 6**

257. `main()`

```
      {
      int a;
      if (a=7)
      printf(" a is 7 ");
      else
      printf("a is not 7");
      }
```

**ans: a is 7**

258. `main()`

```
      {
      int a=4,b=3,c=5;
      if (a>b)
      if(b>c)
      printf("inner");
      else printf("outer");
      }
```

**ans: outer (else is attached to inner if)**

259. `main()`

```
      {
      int a=2,b=3,c=5;
      if (a>b)
      if(b>c)
      printf("inner");
      else printf("outer");
      }
```

**ans: no output (else is attached to inner if)**

260. `main()`

```
      {
```

```
inc(); inc();  inc();
}
inc()
{
static int x;
printf("%d", ++x);
}
```

**ans: 123**

261. main()
```
{
printf("%d", strlen(""));
}
```

**ans: 0 (strlen excludes null character. It is a null string)**

262. main()
```
{
printf("%d", sizeof(""));
}
```

**ans: 1 (sizeof included null character. It is a null string)**

263. main()
```
{
int a=5,b=2;
printf("%d", a+++b);
}
```

**ans: 7**

264. main()
```
{
int v=3, *pv=&v;
printf(" %d %d ", v,*pv);
}
```

**ans: 3 3**

265. main()
```
{
enum cities{bethlehem,jericho,nazareth=1,jerusalem};
printf("%d %d",jericho,nazareth);
}
```

**ans: 1 1**

266. difference between scanf and sscanf function

**ans: sscanf(s,...) is equivalent to scanf(...) except that**
**input charecter are taken from string s.**

267. main()
```
{
char line[80];
scanf("%[^\n]",line);
printf("%s",line);
}
```

**ans: if you type this is manu<enter> output will be this is manu**
**scanf normally takes a single string but if we use [^\n] it takes multiple strings till it encounters newline (i.e., enter is pressed)**

268. main()
```
{
char line[80];
scanf("%[^a]",line);
printf("%s",line);
}
```

**ans: type this is manu<enter> output will be this is m**

269. main()
```
{
char line[80];
scanf("%[^u]",line);
printf("%s",line);
}
```

**ans: type this is manu<enter> output will be this is man**

270. main()
```
{
printf("%f %f",floor(-2.8),ceil(-2.8));
}
```

**ans: -3.000000 -2.000000**

271. int x[3][4] ={
```
{1,2,3},
{4,5,6},
{7,8,9}
}
```

**ans: values in fourth column are zero**

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

        ans: 2

273.  main()
        {
        int a,b;
        int *p,*q;
        a=10;b=19;
        p=&(a+b);
        q=&max;
        }

        ans: error (must take address of memory location)

274.  main()
        {
        printf("%u", sizeof(func));
        }

   func()
        {
        return 0;
        }

        ans: error (sizeof operator operand should not be
        function name)

275.  main()
        {
        printf("%u", sizeof(func()));
        }

   func()
        {
        return 0;
        }

        ans: 2 (sizeof operator operand should not be function
        name but it can be a function call)
```

276.  sizeof operator is **runtime operator**

277.  An array whose elements are fn pointers which inturn returns a
         character

```
            ans: char (*x[]) ();

278.  main()
            {
            int n,i=1;
            switch(n)
            {
            case 1:
            printf("1");
            case 2:
            printf("2");
            default:
            i=10;
            }
            printf("i=%d",i);
            }

            ans: 10 (since n is not initialized it contains garbage
            value hence almost all the times default case is run)

279.  #define max 10
      main()
            {
            int a,b;
            int *p,*q;
            a=10;b=19;
            p=&(a+b);
            q=&max;
            }

            ans: error (must take address of a memory location)

280.  main()
            {
            int i;
            printf("%d", &i)+1;
            scanf("%d", i)-1;
            }

            ans: address of memory location i (scanf function reads
            value into a garbage location if it fall in   protected
            memory it gives error otherwise value will be read into
            that location)

281.  main()
            {
            int i;
            float *pf;
            pf = (float *)&i;
            *pf = 100.00;
            printf("%d", i);
```

```
        }

        ans: runtime error


282.  main()
        {
        int i = 0xff;
        printf("%d", i<<2);
        }

        ans: 1020


283.  #define SQR(x) x * x
      main()
        {
        printf("%d", 225/SQR(15));
        }

        ans: 225


284.  union u
        {
        struct st
        {
        int i : 4;
        int j : 4;
        int k : 4;
        int l;
        }st;
        int i;
        }u;


      main()
        {
        u.i = 100;
        printf("%d, %d, %d",u.i, u.st.i, u.st.l);
        }

        ans: 100 4 0


285.  union x
        {
        union u
        {
        int i;
        int j;
        }a[10];
        int b[10];
        }u;


      main()
```

```
            {
            printf("%d ", sizeof(u));
            printf("%d ", sizeof(u.a));
            printf("%d", sizeof(u.a[0].i));
            }
```

**ans: 20 20 2 (Note: when unions or structures are nested
inner and outer tagnames should be different)**

286. main()
```
            {
            int (*functable[2])(char *format, ...) ={printf, scanf};
            int i = 100;
            (*functable[0])("%d ", i);
            (*functable[1])("%d ", i);
            (*functable[1])("%d ", i);
            (*functable[0])("%d", &i);
            }
```

**ans: runtime error (& is missing)**

287. main()
```
            {
            int (*functable[2])(char *format, ...) ={printf, scanf};
            int i = 100;
            (*functable[0])("%d, ", i);
            (*functable[1])("%d", &i);
            (*functable[1])("%d", &i);
            (*functable[0])(", %d", &i);
            }
```

**ans: 100, enter two values for scanf, i address value.
In function pointers all the functions will have the
same return type.**

288. main()
```
            {
            int i, j, *p;
            i = 25;
            j = 100;
            p = &i; /* Address of i is assigned to pointer p */
            printf("%f", i/(*p)); /* i is divided by pointer p */
            }
```

**ans: runtime error (format specifier %f is not matched)**

289. main()
```
            {
            char *p = "hello world";
            p[0] = 'H';
            printf("%s", p);
```

```
        }

        ans: Hello world
```

290.  main()
```
        {
        char * strA;
        char * strB = "I am OK";
        memcpy( strA, strB, 6);
        }
```

**ans: error (pointer should be initialized before using)**

291.  How will you print % character?

**ans:  printf("\%");  printf("%%");  printf("\%%");**

292.  main()
```
        {
        printf("\% ");
        printf("\\% ");
        printf("%% ");
        printf("\%%");
        }
```

**ans: % \% % %**

293.  main()
```
        {
        printf("\%d ", 100);
        printf("\\% ");
        printf("%% ");
        printf("\%%");
        }
```

**ans: 100 \% % %**

294.  const int perplexed = 2;
```
      #define perplexed 3
      main()
        {
        #ifdef perplexed
        #undef perplexed
        #define perplexed 4
        #endif
        printf("%d",perplexed);
        }
```

**ans: 4 (const int perplexed will not come into picture
bcoz text replacement is done at preprocessor stage**

**which is first stage in executable file development stages)**

295. struct Foo

```
{
char *pName;
};

main()
{
struct Foo *obj = malloc(sizeof(struct Foo));
strcpy(obj->pName,"Your Name");
printf("%s", obj->pName);
}
```

**ans: runtime error (Note: pName should be initialize before using)**

296. struct Foo

```
{
char *pName;
char *pAddress;
};
main()
{
struct Foo *obj = malloc(sizeof(struct Foo));
obj->pName = malloc(100);
obj->pAddress = malloc(100);
strcpy(obj->pName,"Your Name");
strcpy(obj->pAddress, "Your Address");
free(obj);
printf("%s ", obj->pName);
printf("%s", obj->pAddress);
free(obj->pName);
free(obj->pAddress);
}
```

**ans: :Your Name Your Address**

297. main()

```
{
char *a = "Hello ";
char *b = "World";
printf("%s", stract(a,b));
}
```

**ans: stract function should be defined or strcat should be used**

298. main()

```
{
```

```
char *a = "Hello ";
char *b = "World";
printf("%s", strcat(a,b));
}
```

**ans: HelloWorld**

299. main()
```
{
char *a = "";
char *b = "World";
printf("%s", strcpy(a,b));
}
```

**ans: World**

300. void func1(int (*a)[10])
```
{
printf("Ok it works ");
}
```

void func2(int a[][10])
```
{
printf("Will this work?");
}
```

main()
```
{
int a[10][10];
func1(a);
func2(a);
}
```

**ans: Ok it works Will this work?**
**Formal argument in function definition should be a**
**pointer to array or double dimensional array but not a**
**pointer to pointer (doble pointer)**

301. main()
```
{
printf("%d, %d", sizeof('c'), sizeof(100));
}
```

**ans: 2, 2**

302. main()
```
{
int i = 100;
printf("%d", sizeof(sizeof(i)));
}
```

```
        ans: 2

303.  int f();
      main()
          {
          int c = 5;
          printf("%p %p %d %d", f,f(),f,f());
          }
      int f()
          {}

          ans: segment:offset segment:offset integer integer (all
          are unknown values. Segment and offset values of
          function address and function return value.  Values of
          function address and function return value)

304.  main()
          {
          char c;
          int i = 456;
          c = i;
          printf("%d", c);
          }

          ans: -56

305.  main ()
          {
          int x = 10;
          printf ("x = %d, y = %d", x,--x++);
          }

          ans: lvalue required

306.  main()
          {
          int i =10, j = 20;
          printf("%d, %d, ", j-- , --i);
          printf("%d, %d", j++ , ++i);
          }

          ans: 20, 9, 19, 10

307.  main()
          {
          int x=5;
          for(;x==0;x--)
          {
          printf("x=%d\n", x--);
          }
          }
```

ans: no output

308. main()
```
{
int x=5;
for(;x!=0;x--)
{
printf("x=%d ", x--);
}
}
```

ans: infinite loop (becareful here two decrements, and x
is odd. So   x==0 never occurs)

309. main()
```
{
int x=4;
for(;x==0;x--)
{
printf("x=%d ", x--);
}
}
```

ans: x=4 x=2

310. main()
```
{
int x=5;
{
printf("x=%d", x--);
}
}
```

ans: x=5

311. main()
```
{
unsigned int bit=256;
printf("%d ", bit);
{
unsigned int bit=512;
printf("%d", bit);
}
}
```

ans: 256 512

312. main()
```
{
int i;
```

```
for(i=0;i<5;i++)
{
printf("%d ", 1L << i);
}
}
```

**ans: 1 2 4 8 16**

313.  main()
```
{
signed int bit=512, i=5;
for(;i;i--)
{
printf("%d ", bit = (bit >> (i - (i -1))));
}
}
```

**ans: 256 128 64 32 16**

314.  main()
```
{
signed int bit=512, i=5;
for(;i;i--)
{
printf("%d ", bit >> (i - (i -1)));
}
}
```

**ans: 256 256 256 256 256**

315.  main()
```
{
if (!(1&&0))
{
printf("OK I am done.");
}
else
{
printf("OK I am gone.");
}
}
```

**ans: OK I am done**

316.  main()
```
{
if ((1||0) && (0||1))
{
printf("OK I am done.");
}
else
```

```
        {
        printf("OK I am gone."); }
        }
```

**ans: OK I am done**

317. main()
```
        {
        signed int bit=512, mBit;
        {
        mBit = ~bit;
        bit = bit & ~bit ;
        printf("%d %d", bit, mBit);
        }
        }
```

**ans: 0 -513**

318. What is the difference between the following
    a. i=i+1;
    b. ++i;

    **ans: ++i is a single instruction while in i=i+1, first
    i+1 is computed and then assigned.**

319. What is exception handling and how is it different from error
        handling...... Why
    is  exception handling used instead of error handling in some
        cases and vice versa.

320. Explanation of OOP principles
        -Data Abstraction.
        -Data Encapsulation
        -Inheritence
        -Polymorphism
        -Dynamic Binding.
        -Reduction of Errors.

321. main()
```
        {
        int d,a=5,b=3,c=(a,b);
        d=(a,b);
        printf("%d %d",c,d);
        }
```

    **ans: 3 3 (from 321 to 324 think about comma operator)**

322. main()
```
        {
        int a=5,b=3,c=a,d;
        d=(a,b);
```

```
        printf("%d %d",c,d);
        }

        ans: 5 3

323.  main()
        {
        int a=5,b=3,c=(a,b),d;
        d=(a,b);
        printf("%d %d",c,d);
        }

        ans: 3 3

324.  main()
        {
        int a=5,b=3,c=(a,b),d;
        d=a,b;
        printf("%d %d",c,d);
        }

        ans: 3 5 (from 321 to 324 think about comma operator)

325.  Which one is having problem?
      int *f1()
        {
        int n;
        return (n)
        }

      int *f2()
        {
        int *p;
        *p=3;
        return p;
        }

      int *f3()
        {
        int *p;
        p=malloc();
        return p;
        }

      int *f4()
        {
        int n;
        return (&n)
        }
```

**ans: f4 is having problem as it is returning address of
        auto variable.**

326. `*p+=1`
     `*p++`
     are these two same?

        **ans: not same (first one increments value pointed by p
        and second one   increments pointer)**

327. `int num[3];`
     `num[3]=2;`

        **ans: array index exceeds array bounds**

328. 
```
main()
        {
        int j=4;
        for(int i=0;i<5;i++)
        {
        j++;
        ++j;
        }
        printf("%d",j);
        }
```

        **ans: undefined symbol i**

329. 
```
main()
        {
        int j=4;
        for(int i=0;i<5;i++)
        {
        j++;
        ++j;
        }
        printf("%d",j);
        }
```

        **ans: 14**

330. 
```
main()
        {
        char s1[20]="hello world";
        s1[5]="\0";
        printf("%d",strlen(s1));
        }
```

        **ans: nonportable pointer conversion**

331. `main()`

```
{
char s1[20]="hello world";
s1[5]='\0';
printf("%d",strlen(s1));
}
```

**ans: 5**

332.  Which can't be passed to subroutine

**ans:preprocessor directive.**

333.  #define m **10**
      **f();**
      main()
```
{
f(m);
}
```
      **f(int j) or f(j)**
```
{
printf("%d",j);
}
```

**ans: 10**

334.  #define m **10.0**
      **f(float);**
      main()
```
{
f(m);
}
```
      **f(float j)**
```
{
printf("%f",j);
}
```

**ans: 10.000000 (careful about macro value type and proceed)**

335.  **f();**
      main()
```
{
int x=1,y=2,z=3;
f(x,y,z);
}
```
      **f(int p,int q,int r)**
```
{
printf("%d %d %d",p,q,r);
}
```

```
            ans: 1 2 3 (in prototype we have not given argument
            types as they are ints)

336.  f();
      main()
            {
            float x=1.0,y=2.0,z=3.0;
            f(x,y,z);
            }
      f(float p,float q,float r)
            {
            printf("%f %f %f",p,q,r);
            }

            ans: error (no prototype)

337.  f(float, float, float);
      main()
            {
            float x=1.0,y=2.0,z=3.0;
            f(x,y,z);
            }
      f(float p,float q,float r)
            {
            printf("%f %f %f",p,q,r);
            }

            ans: 1.000000 2.000000 3.000000

338.  main()
            {
            int x=0;
            for(;;x++){
            if(x==4) break;
            continue;
            }
            printf("%d\n",x);
            }

            ans: 4

339.  main()
            {
            int i=100;
            do
            {--i;}while(i>50);
            printf("%d\n",i);
            }

            ans: 50
```

```
340. main()
        {
        int o;
        int m=-14;
        int n=6;
        o=m%++n;
        n+=m++%o;
        printf("%d%d%d",m,n,o);
        }
```

**ans: divide by zero error**

```
341. main()
        {
        int a=1000,b=1000,c;
        (long)c=(long)a*b;
        printf("%d",c);
        }
```

**ans: error (lvalue required)**

342. Debugging is the process of finding

**ans : logical and runtime errors**

343. using ternary find out max of a,b,c

**ans: (a>b) ? (a>c ? a : c) : (b>c ? b : c)**

```
344. main()
        {
        int a, *b = &a, **c =&b;
        a=4;
        ** c= 5;
        printf("%d",a);
        }
```

**ans: 5**

```
345. main( )
        {
        int i = 1;
        if(!i)
        printf("Recursive calls are real pain!");
        else
        {
        i = 0;
        printf("Recursive calls are challenging\n");
        main();
        }
        }
```

**ans: prints Recursive calls are challenging infinite times or till stack overflows.**

346. main()

```
{
struct emp{
char n[20];
int age;};
struct emp e1={"david",23};
struct emp e2=e1;
if(e1==e2)
printf("structures are equal");
}
```

**ans: structures are equal (in ANSI C) but error in some other compilers. Direct assignment and comparisons can't be done.**

347. main( )

```
{
char a[];
a[0] = 'A';
printf("%c", a[0]);
}
```

**ans: size of a is unknown**

348. main()

```
{
printf("%d %d %d",sizeof('3'),sizeof("3"),sizeof(3));
}
```

**ans: 2 2 2**

349. main()

```
{
printf("%c","abcdefgh"[4]);
}
```

**ans: e**

350. main()

```
{
int a[ ]={10,20,30,40,50};
char *p;
p=(char *)a;
printf("%d",*((int *)p+4));
}
```

**ans: 50**

```
351.  main()
      {
      int a[]={10,20,30,40,50};
      char *p;
      p=(char *)a;
      printf("%d %d %d %d",*p,*(p+1),*(p+2),*(p+3));
      }
```

**ans: 10 0 20 0**

```
352.  main()
      {
      printf("%c",7["sundaram"]);
      }
```

**ans: m**

```
353.  #define str(x) #x
      #define Xstr(x) str(x)
      #define oper multiply
      main()
      {
      char *opername=Xstr(oper); /* #multiply  i.e.,
      "multiply"
      printf("%s",opername);
      }
```

**ans: multiply (#, stringizing operator allows a formal argument within a macro definition to be converted to a string)**

```
354.  #define sqr(x) (x*x)
      main()
      {
      int a,b=3;
      a=sqr(b+2);
      printf("%d",a);
      }
```

**ans: 11**

```
355.  main()
      {
      int b;
      b=f(20);
      printf("%d",b);
      }
      f(int a)
      {
      a>20 ? return (10): return (20);
```

```
        }
            ans: error in function definition

356.  main()
            {
            int b;
            b=f(20);
            printf("%d",b);
            }
      f(int a)
            {
            return a>20 ? (10): (20);
            }

            ans: 20

357.  What error would the following function give on compilation.
      f(int a,int b)
      {
      int a;
      a=20;
      return a;
      }
            ans: redeclaration of a

358.  main()
            {
            int i=3;
            i=i++;
            printf("%d",i);
            }

            ans: 4

359.  main()
            {
            static char a[]="Bombay";
            char *b="Bombay";
            printf("%d %d",sizeof(a),sizeof(b));
            }

            ans: 7 4 (here pointer takes 4 bytes)

360.
      main()
            {
            int x = 5;
            printf("%d %d", x++, ++x);
            return 0;
            }
```

```
              ans: 6 6

361.  main()
          {
          int z = 4;
          printf("%d", printf(" %d %d ", z, z));
          }

          ans: 4 4 5 (three spaces are there total five characters
          will be printed by printf statement)

362.  main()
          {
          int z = 45;
          printf("%d", printf(" %d %d ", z, z));
          }

          ans: 45 45 7

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

          ans: lvalue required

364.  main()
          {
          int n=20, i = 0;
          while(n-->0);
          i = i+n;
          printf("%d",i);
          }

          ans: -1

365.  main()
          {
          int i = 0; char ch = 'A'
          do {
          printf("%c", ch);
          } while (i++ <5| | ++ch < ='F');
          }
```

**ans: AAAAAABCDEF**

366. int count, sum;
```
        main()
        {
        for(count = 4; sum += --count; );
        printf("%d", sum);
        }
```

**ans: 0**

367. main( )
```
        {
        static float a[ ] = { 13.24, 1.5}
        float *j, *k;
        j = a;
        k = a + 2;
        j = j * 2;
        k = k/2;
        printf("%f%f ", *j, *k);
        }
```

**ans: error (pointer multiplication and division is illegal)**

368. main( )
```
        {
        static char s[ ] = "Rendezvous";
        printf("%d", *(s+ strlen(s)));
        }
```

**ans: 0**

369. main()
```
        {
        char **p="Hello";
        printf("%c",*p);
        }
```

**ans: H**

370. main()
```
        {
        char **p="Hello";
        printf("%s",p);
        }
```

**ans: Hello**

371. main()
```
        {
```

```
        char **p="Hello";
        printf("%s",*p); /* (or) printf("%s",**p); */
        }

        ans: error
```

372. main()
```
        {
        char **p="Hello";
        printf("%c",**p);
        }

        ans: error
```

373. main()
```
        {
        char a[]="Hello";
        printf("%c\n",*a++);
        }

        ans: lvalue required
```

374. main()
```
        {
        int a=3,b=2,c=1;
        static int k= a<b<c-1;
        printf("%d",k);
        }

        ans: illegal initialization (for static initializer
        should be constant expression or constant)
```

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

        ans: 0
```

376. main()
```
        {
        char c=-32;
        int i=-64;
        unsigned u=-26;
        if(c>i)
        printf("PASS1 ");
        if( i < c)
        printf("PASS2 ");
        else
```

```
                printf("FAIL1 ");
                if(i<u)
                printf("PASS2 ");
                else
                printf("FAIL2 ");
                }
```

**ans: PASS1 PASS2 PASS2**

377.  main()
```
                {
                int i=4;
                switch(i)
                {
                case 1:
                printf("HEllo");
                case default: // "case" should not come with "default"
                printf("****");
                }
                }
```

**ans: error (case should not be there with default)**

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

**ans: 5 4 3 2 1**

379.  main()
```
                {
                int a=5,c;
                int ptr;
                ptr=&a;
                c=*ptr * a;
                printf("%d,%d",c,a);
                }
```

**ans: error (nonportable pointer conversion and invalid
indirection)**

380.  main()
```
                {
                int x=10,y=5,p,q;
                p=x>9;
                q=x>3&&y!=3;
                printf("p=%d q=%d",p,q);
```

```
            }
```

**ans: p=1 q=1**

381.  main()
```
            {
            int x=11,y=6,z;
            z=x==5||y!=4;
            printf("z=%d",z);
            }
```

**ans: z=1**

382.  main()
```
            {
            int c=0,d=5,e=10,a;
            a=c>1?d>1||e>1?100:200:300;
            printf("a=%d",a);
            }
```

**ans: a=300**

383.  main()
```
            {
            int i=-5,j=-2;
            junk(i,&j);
            printf("i=%d,j=%d",i,j);
            }
            junk(i,j)
            int i,*j;
            {
            i=i*i;
            *j=*j**j;
            }
```

**ans: i=-5,j=4**

384.  #define NO
      #define YES
      main()
```
            {
            int i=5,j;
            if(i>5)
            j=YES;
            else
            j=NO;
            printf("%d",j);
            }
```

**ans: error (NO and YES are not defined)**

```
385.  #define NO 0
      #define YES 1
      main()
              {
              int i=5,j;
              if(i>5)
              j=YES;
              else
              j=NO;
              printf("%d",j);
              }
```

**ans: 0**

```
386.  main()
              {
              int a=0xff;
              if(a<<4>>12)
              printf("leftist");
              else
              printf("rightist");
              }
```

**ans: rightist**

```
387.  main()
              {
              int i=+1;
              while(~i)
              printf("vicious circles");
              }
```

**ans: infinite loop**

388.  What's the use of sizeof( ) function... since one can
      always directly write number of bytes instead of
      calling the function.

**ans: for runtime operations**

```
389.  main()
              {
              int p = -200;
              char c;
              c = p;
              printf("%d %d", c++, ++c);
              }
```

**ans: 57 57**

```
390.  int a=1;
```

```
int ab=4;
int main()
    {
    int b=3,a=2;
    printf("%i*/%i*/%*/i",a,b,ab);
    }
```

**ans: 2*/3*/%*/i**

391. Which one of the following statements allocates enough space
     to hold an array of 10 integers that are initialized to 0 ?

**ans: int *ptr = (int *) calloc(10,sizeof(int));**

392. main()
```
    {
    int i,j;
    j = 10;
    i = j++ - j++;
    printf("%d %d", i,j);
    }
```

**ans: 0 12**

393. main()
```
    {
    int j;
    for(j=0;j<3;j++)
    foo();
    }
    foo() {
    static int i = 10;
    i+=10;
    printf("%d ",i);
    }
```

**ans: 20 30 40**

394. What is wrong in the following code
     main()
```
    {
    char *c;
    c = "Hello";
    printf("%s\n", c);
    }
```

**ans: Hello (nothing wrong with the code)**

395. main()
```
    {
```

```
        union {
        int a;
        int b;
        int c;
        } u,v;
        u.a = 10;
        u.b = 20;
        printf("%d %d \n",u.a,u.b);
        }
```

**ans: 20 20**

396. main()
```
        {
        char *str = "12345";
        printf("%c %c %c\n", *str, *(str++), *(str++));
        }
```

**ans: 3 2 1**

397. #define max(a,b) (ab)?a:b
main()
```
        {
        int a,b;
        a=3;
        b=4;
        printf("%d",max(a,b));
        }
```

**ans: error (undefined symbol ab when it is replaced in printf statement)**

398. main()
```
        {
        int len=4;
        char *st="12345678";
        st = st -len;
        printf("%c\n",*st);
        }
```

**ans: some junk character is printed**

399. func();
main()
```
        {
        func(1);
        }
        func(int i)
        {
        static char *str ={ "One","Two","Three","Four"};
        printf("%s\n",str[i++]);
```

```
            return;
            }

        ans: error in declaration and definition. Pointer should
        be there
```

400. main()
```
            {
            int i;
            for (i=1;i<100; i++)
            printf("%d %0x\n",i,i);
            }
```

        **ans: 1 to 99 will be printed both in decimal and
        hexadecimal form**

401. struct {
```
     int x;
     int y;
     union {
     int id_no;
     char *name;
     }b;
     }s,*st;
     main()
            {
            st = &s;
            st-x=10;
            st-b.id_no = 101;
            printf("%d %d\n",s.x,s.b.id_no);
            }
```

        **ans: error (undefined symbol i and b.  i and b should
        not be used as direct variables. They should be
        associated with structure variable)**

402. main()
```
            {
            int j,ans;
            j = 4;
            ans = count(4);
            printf("%d\n",ans);
            }
     int count(int i)
            {
            if ( i < 0) return(i);
            else
            return( count(i-2) + count(i-1));
            }
```

        **ans: -18**

```
403. main()
        {
        int i=4;
        if(i=0)
        printf("statement 1");
        else
        printf("statement 2");
        }
```

**ans: statement 2**

```
404. main()
        {
        char a[2];
        *a[0]=7;
        *a[1]=5;
        printf("%d",&a[1]-a);
        }
```

**ans: invalid indirection**

```
405. main()
        {
        char a[]="hellow";
        char *b="hellow";
        char c[5]="hellow";
        printf("%s %s %s ",a,b,c);
        printf("%d %d %d",sizeof(a),sizeof(b),sizeof(c));
        }
```

**ans: too many initializers (c array size is less)**

```
406. main()
        {
        char a[]="hellow";
        char *b="hellow";
        char c[7]="hellow";
        printf("%s %s %s ",a,b,c);
        printf("%d %d %d",sizeof(a),sizeof(b),sizeof(c));
        }
```

**ans: hellow hellow hellow 7 4 7 (here pointer takes 4 bytes)**

```
407. int num[]={10,1,5,22,90};
     main()
        {
        int *p,*q;
        int i;
        p=num;
```

```
q=num+2;
i=*p++;
printf("%d %d",i,p-q);
}
```

**ans: 10 -1**

408. One pointer declaration is given like this:
     `int *(*p[10])(char *, char*)`
     Explain the variable assignment

     **ans: an array of 10 pointers to functions with two
     character pointers as arguments and returning integer
     pointer.**

409. ```
     main()
     {
     char *a[4]={"jaya","mahe","chandra","buchi"};
     printf("%d %d %d",sizeof(a),sizeof(char
     *),sizeof(a)/sizeof(char *));
     }
     ```

     **ans: 16 4 4 (pointer takes 4 bytes)**

410. The integers from 1 to n are stored in an array in a random
     fashion. but one integer is
     missing. Write a program to find the missing integer.

     **ans: The sum of n natural numbers is = n(n+1)/2.
                  if we subtract the above sum from the sum of
     all the
                  numbers in the array , the result is nothing
     but the
                  missing number.**

411. Write a C program to find whether a stack is progressing in
     forward or reverse direction.

412. Write a C program that reverses the linked list.

413. ```
     #define MAX(x,y) ((x)>(y)?(x):(y))
     main()
     {
     int x=5,y=5;
     printf("maximum is %d",MAX(++x,++y));
     }
     ```

     **ans: maximum is 7 (careful about braces not only in
     printf but also in macro definition.**

414. main()

```
      {
      int *p,*q,r;
      int values[30];
      p=&values[0];
      q=values+29;
      r=++q-p;
      printf("%d",r);
      }
```

**ans: 30**

415.  static int i = 5;
      main()
```
      {
      int sum=0;
      do
      {
      sum +=(1/i);
      }while(0<i--);
      }
```

**ans: error (divide by zero)**

416.  enum mode = {green,red,orange,blue ,white};
      main ()
```
      {
      green = green +1;
      printf("%d,%d",green,red );
      }
```

**ans: error (lvalue required since green is a symbolic constant and = operator should not be there in enum declaration)**

417.  int (*(*ptr)(int)) (void)

**ans: ptr is pointer to function that takes an int value returns a pointer to a function with a no argument which returns a integer**

418.  char *c[] ={
              "FILE",
              "EDIT",
              "SEARCH",
              "COMPILE"
          };

      char **cp[] = {c+3,c+2,c+1,c};
      char ***cpp = cp;
      main()

```
        {
        printf("%s ", **cpp);
        printf("%s", *--*++cpp+3);
        printf("%s", *cpp[-2]+3);
        printf("%s\n",cpp[-1][-1]+1);
        }
```

**ans: COMPILE T (last two printfs cause error)**

419.  struct x
      {
      int j;
      char k[100];
      unsigned i;
      };
      int *ptr1;
      struct X *ptr2;

      main()
          {
          printf("%d %d",sizeof(ptr1),sizeof(ptr2));
          }

      **ans: 4 4**

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

      **ans: 7 6 5**

421.  main()
          {
          int i,j ;
          for(i=0;i<=10;i++);
          for(j=0;j<=10;j++);
          printf("i=%d,j=%d\n",i,j);
          }

      **ans: i=11,j=11**

422.  #define square(a) (a*a)
      main()
          {
          printf("%d",square(4+5));
          }

      **ans: 29**

```
423.  main()
          {
          int p = 0, q =1;
          p = q++;
          p = ++q;
          p = q--;
          p = --q;
          printf("%d %d",p,q);
          }

          ans: 1 1

424.  main()
          {
          int a , count;
          int func(int);
          for (count = 1 ;count <=5;++count)
          {
          a = func(count);
          printf("%d", a);
          }
          }
      int func(int x)
          {
          int y;
          y=x*x;
          return(y);
          }

          ans: 1491625

425.  supposing that each integer occupies 4 bytes and each
          charactrer 1 byte , what is the
      output of the following programme?

      main()
          {
          int a[] ={ 1,2,3,4,5,6,7};
          char c[] = {'a','x','h','o','k'};
          printf("%d %d", (&a[3]-&a[0]),(&c[3]- &c[0]));
          }

          ans: 3 3

426.  main()
          {
          struct s1 {int i; };
          struct s2 {int i; };
          struct s1 st1;
          struct s2 st2;
          st1.i =5;
```

```
            st2 = st1;
            printf(" %d " , st2.i);
            }
```

**ans: error (different struct variables should not
assigned using "=" operator.)**

427. ```
     main()
            {
            int i,j;
            int mat[3][3] ={1,2,3,4,5,6,7,8,9};
            for (i=2;i>=0;i--)
            for (j=2;j>=0;j--)
            printf("%d" , *(*(mat+j)+i));
            }
     ```

     **ans: 963852741**

428. ```
     main()
            {
            int n=10;
            fun(n);
            }
     int fun( int n)
            {
            int i;
            for(i=0;i<=n;i++)
            fun(n-i);
            printf(" well done");
            }
     howmany times is the printf statement executed for n=10?
     ```

     **ans:** Before reaching to printf statement it will goes to
     infinite loop.

429. ```
     main()
            {
            struct emp{
            char emp[];
            int empno;
            float sal;
            };
            struct emp member = { "TIGER"};
            printf(" %d %f", member.empno,member.sal);
     ```

     **ans: error(array size is not declared if it is declared
     ans is 0 0.000000)**

430. ```
     # define infiniteloop while(1)
     main()
     ```

```
        {
        infiniteloop;
        printf("DONE");
        }
```

**ans: infiniteloop in main ends with ";" . so loop will not reach end;and the DONE also will not print.**

431.  main()
```
        {
        int a=2, b=3;
        printf(" %d ", a+++b);
        }
```

**ans: 5**

432.  #define prn(a) printf("%d ",a)
      #define print(a,b,c) prn(a), prn(b), prn(c)
      #define max(a,b) (a<b)? b:a

      main()
```
        {
        int x=1, y=2;
        print(max(x++,y),x,y);
        print(max(x++,y),x,y);
        }
```

**ans: 2 2 2 3 4 2**

433.  #define PRINT(int) printf("int=%d ",int);
      main()
```
        {
        int x,y,z;
        x=03;y=-1;z=01;
        PRINT(x^x);
        z<<=3;PRINT(z);
        y>>=3;PRINT(y);
        }
```

**ans: int=0 int=8 int=-1**

434.  main()
```
        {
        int i;
        i=1;
        i=i+2*i++;
        printf("%d",i);
        }
```

**ans: 4**

```
435.  main()
          {
          char ch='A';
          while(ch<='F')
          {
          switch(ch)
          {
          case'A':case'B':case'C':case'D':ch++;continue;
          case'E':case'F':ch++;
          }
          putchar(ch);
          }
          }
```

**ans: FG**

```
436.  main()
          {
          int a=1, b=2, c=3, *pointer;
          pointer=&c;
          a=c/*pointer;
          b=c;
          printf ("a=%d b=%d",a,b);
          }
```

**ans: error (there should be space between / and \***
**otherwise it will be starting of comment)**

```
437.  #define MAN(x,y) (x)>(y)?(x):(y)
      main()
          {
          int i=10,j=5,k=0;
          k= MAN(i++,++j);
          printf("%d %d %d %d",i,j,k);
          }
```

**ans: 12 6 11 garbage value**

```
438.  main()
          {
          int a=10,b=5, c=3,d=3;
          if(a<b)&&(c=d++)
          printf("%d %d %d %d" ,a,b,c,d);
          else printf("%d %d %d %d", a,b,c,d);
          }
```

**ans: error (if condition should be parenthesis)**

```
439.  main(int size of arg ,char *arg[])
          {
          while(size of arg)
```

```
         printf("%s",arg[--size of arg]);
         }
```

**ans: error (no space between sizeofarg)**

440. main(int sizeofarg ,char *arg[])
```
         {
         while(sizeofarg)
         printf("%s",arg[--sizeofarg]);
         }
```

**ans: f:\progr.exe**

441. main()
```
         {
         int i=3;
         while(i--)
         {
         int i=100;
         i--;
         printf("%d..",i);
         }
         }
```

**ans: 99..99..99..**

442. main()
```
         {
         int rows=3,colums=4;
         int a[rows][colums]={1,2,3,4,5,6,7,8,9,10,11,12};
         int i, j,k; i=j=k=99;
         for(i=0;i<rows;i++)
         for(j=0;j<colums;j++)
         if(a[k][j]<k) k=a[i][j];
         printf("%d\n",k);
         }
```

**ans: error (constant expression required in array dimension)**

443. main()
```
         {
         int x=10,y=15;
         x=x++;
         y=++y;
         printf("%d %d\n",x,y);
         }
```

**ans: 11 16**

444. main()

```
{
int x=20,y=35;
x = y++ + x++;
y = ++y + ++x;
printf("%d %d\n",x,y);
}
```

**ans: 57 94**

445.  main()
```
{
char *p1="Name";
char *p2;
p2=(char *)malloc(20);
while(*p2++=*p1++);
printf("%s\n",p2);
}
```

**ans: unknown string will be printed pointer p2 points to next character to null character.**

446.  main()
```
{
int x=5;
printf("%d %d %d\n",x,x<<2,x>>2);
}
```

**ans: 5 20 1**

447.  #define swap1(a,b) a=a+b;b=a-b;a=a-b;
```
main()
{
int x=5,y=10;
swap1(x,y);
printf("%d %d\n",x,y);
swap2(x,y);
printf("%d %d\n",x,y);
}

int swap2(int a,int b)
{
int temp;
temp=a;
b=a;
a=temp;
return;
}
```

**ans: 10 5**
        **10 5 (swap2 won't swap x and y)**

```
448.  main()
        {
        char *ptr = "Ramco Systems";
        (*ptr)++;
        printf("%s\n",ptr);
        ptr++;
        printf("%s\n",ptr);
        }
```

**ans: Samco Systems**
**amco Systems**

```
449.  main()
        {
        char s1[]="Ramco";
        char s2[]="Systems";
        s1=s2;
        printf("%s",s1);
        }
```

**ans: error (lvalue required)**

```
450.  main()
        {
        char *p1;
        char *p2;
        p1=(char *) malloc(25);
        p2=(char *) malloc(25);
        strcpy("Ramco",p1);
        strcpy(p2,"Systems");
        strcat(p1,p2);
        printf("%s",p1);
        }
```

**ans: RamcoSystems**

```
451.  main()
        {
        char a[2];
        *a[0]=7;
        *a[1]=5;
        printf("%d",&a[1]-a);
        }
```

**ans: error (invalid indirection)**

```
452.  main()
        {
        char a[]="hellow";
        char *b="hellow";
        char c[5]="hellow";
```

```
        printf("%s %s %s ",a,b,c);
        printf(" ",sizeof(a),sizeof(b),sizeof(c));
        }
```

**ans: error (Too many initializers)**

453.  main()
```
        {
        char a[]="hellow";
        char *b="hellow";
        char c[7]="hellow";
        printf("%s %s %s ",a,b,c);
        printf("%d %d %d ",sizeof(a),sizeof(b),sizeof(c));
        }
```

**ans: hellow hellow hellow 7 4 7 (pointer takes 4 bytes)**

454.  int a[10]={60,57,10,5,4,3,2,8,9};

     main()
```
        {
        int varx,vary,i;
        for (i=0;i<10;i++)
        {
        if(varx<a[i])
        {
        vary=varx;
        varx=a[1];
        }
        else if (vary<a[i])
        {
        varx=vary;
        vary=a[i];
        }
        printf("%d %d \n",varx,vary);
        }
        }
```

**ans: garbage values of varx and vary are printed 10
times**

455.  #define SWAP(x,y) t=x;x=y;y=t;
     main()
```
        {
        int x=5,y=6;
        if (x>y)
        SWAP(x,y);
        printf("x=%d y=%d\n",x,y);
        }
```

**ans: error (undefined symbol t)**

```
456. main()
        {
        int i=6;
        int j;
        j=sum(i);
        printf("%d",j);
        }
    sum(int x)
        {
        int t;
        if(x<=1) return (1);
        t=sum(x-3)+sum(x-1);
        return (t);
        }

        ans: 9


457. main()
        {
        int a[]={0,2,4,6,8};
        int *ptr;
        ptr=a;
        printf("%d", *((char *) ptr+4));
        }

        ans: 4


458. main()
        {
        int I=3;
        while(I--)
        {int I=100;
        I--;
        printf("%d", I);
        }
        }

        ans: 999999


459. main()
        {
        char ch;
        for(ch='0';ch<=255;ch++)
        printf("%c", ch);
        }

        ans: infinite loop (signed character varies from -128 to
        127)


460. x=3
```

```
function(++x)...value 4 is passed to the function

x=3
function(x++)...value 3 is passed to the function
```

461.  What is runtime locatable code?

What is volatile, register definition in C

What is compiler and what its output.

462.  which of the following is illegal for the program?
```
main()
     {
     char const *p='p';
     }
1)p++ 2) *p++ 3)(*p)++ 4) all
```

**ans: 3 (*p)++ (cannot modify a constant object)**

463.
```
#define putchar(c) printf("%c",c)
main()
     {
     int c='d';
     putchar(c);
     }
```

**ans: d**

464.  void main (void)
```
     {
     printf("%d", printf("ABC\\"));
     }
```

**ans: ABC\4**

465.  void main(void)
```
     {
     int a[10], i;
     int *b;
     b=( int*) malloc(10* sizeof(int)) ;
     *b =&a[3];
     for(i=0;i<10;i++)
     a[i] =i+10;
     printf("%d",b[-1]);
     }
```

**ans: error (nonportable pointer conversion)**

466.  void main(void)

```
{
int a[10], i;
int *b;
b=( int*) malloc(10* sizeof(int)) ;
b =&a[3];
for(i=0;i<10;i++)
a[i] =i+10;
printf("%d",b[-1]);
}
```

**ans: 12**

467. main()
```
{
int a[10]={1,2,3,4,5,6,7,8,9,10};
int *p=a;
int *q=&a[9];
printf("%d",q-p+1);
}
```

**ans: 10**

468. main()
```
{
int i=6;
int *p=&i;
free(p);
printf("%d",i);
}
```

**ans: 6**

469. main()
```
{
int i=5;
i=!i>3;
printf("%d",i);
}
```

**ans: 0**

470. main()
```
{
int a[10];
3[a]=10;
printf("%d",*(a+3));
}
```

**ans: 10**

471. int (*p[10]) ();
```

```
          ans: p is array of pointers that each points to
          a function that takes no arguments and returns
          an int.

472.  struct emp
          {
          int a=25;
          char b[20]="tgk";
          };
      main()
          {
          emp e;
          e.a=2;
          strcpy(e.b,"tellapalli");
          printf("%d %s",e.a,e.b);
          }

          ans: error (structure members should not be initialized
          directly and struct keyword should be there before emp
          e;)

473.  main()
          {
          int a=5;
          const int *p=&a;
          *p=200;
          printf("%d",*p);
          }

          ans: error (cannot modify a constant object)

474.  #define SQ(x) x*x
      main()
          {
          int a=SQ(2+1);
          printf("%d",a);
          }

          ans: 5

475.  main()
          {
          struct t
          {
          int i;
          } a,*p=&a;
          p->i=10;
          printf("%d",(*p).i);
          }
```

**ans: 10**

476. a) for(int i=0; i<50 ;i++)
     for( int j=0; j<100; j++)
     a[i][j]=100;
     b) for(int i=0; i<100 ;i++)
     for( int j=0; j<50; j++)
     a[j][i]=100;

     Which of the above 2 codes executes quickly.

     **ans: a-code takes 5050 comparisons and 5050 increments
     and b-code takes 5100 comparisons and 5100 increments.
     So a-code executes quickly (which is having outer loop
     count less)**

477. i) (*ptr)++;
     ii) *ptr+=1;
     iii) *ptr++;

     which of the folowing is same.

     **ans: i) and ii) are same**

478. void main()
        {
        char *s="susan";
        clrscr();
        printf(s);
        getch();
        }

        **ans: susan**

479. void main()
        {
        int a[20];
        clrscr();
        *a=(int*)malloc(sizeof(a));
        printf("%d",sizeof(a));
        getch();
        }

        **ans: error (nonportable pointer conversion)**

480. void main()
        {
        void fun(int,int);
        int i ,j;
        i=2,j=3;
        fun(i++,j++);

```c
        printf("%d %d",i,j);
        getch();
        }
void fun(int i,int j)
        {
        i++,j++;
        }
```

**ans: 3 4 (no syntax error in function as it is a comma operator)**

481. 
```c
void main()
        {
        int ctr=0;
        clrscr();
        switch(ctr)
        {
        case 0:
        ctr++;
        case 1:
        ctr++;

        default :
        ctr++;
        };
        printf("%d",ctr);
        getch();
        }
```

**ans: 3**

482. 
```c
#define putchar(c) printf("%c",c);
main()
        {
        int c=69;
        putchar(c);
        }
```

**ans: E**

483. 
```c
main()
        {
        printf("%d",printf("ABC//"));
        }
```

**ans: ABC//5**

484. 
```c
main()
        {
        int i=6;
        printf("%d",func(i));
```

```
        }
    int func(int r)
        {
        int static result;
        if(r<=0) result=1;
        else
        result=func(r-3)+func(r-1);
        return result;
        }

        ans: 13
```

485.  main()
```
        {
        int i=3;
        while(i--)
        {
        int i=100;
        i--;
        printf("%d..",i);
        }
        }
```

**ans: 99..99..99..**

486.  #define putchar(c) printf("%c",c)
      void main()
```
        {
        char s='c';
        putchar (s);
        }
```

**ans: c**

487.  **#define putchar (c) printf("%c",c)**
      void main()
```
        {
        char s='c';
        putchar (s);
        }
```

**ans: error (gap should not be there between putchar and (c) )**

488.  void main()
```
        {
        int a[]={9,4,1,7,5};
        int *p;
        p=&a[3];
        printf("%d",p[-1]);
        }
```

ans: 1

489. ```c
void main()
    {
    int a[]={10,20,30,40,50};
    int *p;
    p= (int*)((char *)a + sizeof(int));
    printf("%d",*p);
    }
```

ans: 20

490. Which code will run faster

```c
for(i=0;i<100;i++)
for(j=0;j<10;j++)
a[i][j]=0;
```

OR

```c
for(j=0;j<10;j++)
for(i=0;i<100;i++)
a[i][j]=0;
```

ans: first code (1100 increments 1100 comparisons)
      second code (1010 increments 1010 comparisons)
    second code will run faster (which is having outer loop
count less)

500. ```c
main()
    {
    void print(int);
    int i=5;
    print(i);
    }
    void print(int n)
    {
    if(n>0)
    {
    print(n-1);
    printf("%d",n);
    print(n-1);
    }
    }
```

ans: 1213121412131215121312141213121

501. ```c
int * f(int a)
    {
    int i;
```

```
    i=a;
    return(&i);
    }
```

**ans: we can't return address of auto variable as it
is allocation is made in stack which is deallocated
when the function returns.**

502. (1)To find string length by using recursive function.
     (2)To find fibonaci series by using recursive
     function.
     (3)To write code for malloc so that allocation may be
     made fastly.
     (4)Write a fn prototype which return a pointer which
     points to an array of 10 ints.

     **ans: int (\*f())[10]**

503. void main ()
```
    {
    int a[]={101,201,301,401,501,601,701,801,901,001};
    int *p; clrscr ();
    printf("%d ",a);
    printf("arthi ");
    printf("%d ", ((char *)a + sizeof(int)));
    p=(int *) ((char *) a +sizeof (int));
    printf("%d",*p);
    }
```

     **ans: 8684 arthi 8686 201 (address of a = 8684)**

504. void main ()
```
    {
    int a[]={101,201,301,401,501,601,701,801,901,001};
    int *p; clrscr ();
    printf("%d ",a);
    printf("arthi ");
    printf("%d ", ((char *)-a + sizeof(int)));
    p=(int *) ((char *) a +sizeof (int));
    printf("%d",*p);
    }
```

     **ans: error (illegal use of pointer)**

505. main ()
```
    {
    int a[10]={10,9,8,7,6,5,4,3,2,1};
    clrscr();
    int *p=a;
    int *q=&a[7];
    printf("%d %d ",q,p);
```

```
            }
```

**ans: error (declaration is not allowed here since clrscr() function is there. Declaration should come before any executable statement)**

506. ```
main()
     {
     printf("%d",printf("HelloSoft"));
     }
```

**ans: HelloSoft9**

507. ```
main()
     {
     int i=3;
     printf("%d %d %d",i++,i,++i);
     }
```

**ans: 4 4 4**

508. ```
main()
     {
     int i=10;
     int j,k=5;
     int a[10];
     for(j=0;j<10;j++)
     a[j]=(i+k)+(i*k);
     }
     Optimize the above code.
```

**ans: main()**
> **{**
> **int i=10,k=5,j,a[10];**
> **for(j=0;j<10;j++)**
> **a[j]=65;**
> **}**

509. ```
main()
     {
     int *p=0x100;
     int *q=0x100;
     int k=p*q;
     printf("%x\n",k);
     }
```

**ans: error (pointer multiplication is not valid)**

510. ```
Char* foo(Str...)
     {
     char str[4];
```

```
        strcpy(str,"HelloSoft");
        return str;
              }
```

**ans: we can't return address of auto variable as it
is allocation is made in stack which is deallocated
when the function returns.**

511.  `int a[10][20][30][40];`
      `int *p`
      How to access an element of a using p?

      **ans: a[i][j][k][l]        *(p+**

512.  `main()`
```
        {
        int i=10;
        if(i>20)
        if(i==10)
        printf("Hi");
        else
        printf("Bye");
        }
```

      **ans: no output**

513.  If a row daminated two dimentional arry in the following which
      one is advantage
      and why?

      ```
      a) for(i=0;i<1000;i++)
      for(j=0;j<1000;j++)
      temp=temp+a[i][j];
      ```


      ```
      b) for(j=0;j<1000;j++)
      for(i=0;i<1000;i++)
      temp=temp+a[i][j]
      ```

      **ans: a (just it is a guess. In 'a' we are accessing
      elements which are in adjacent locations. In 'b' we are
      accessing elements which are 1000 locations apart)**

514.  `void main()`
```
        {
        printf("%d",(float)3/2);
        }
```

      **ans: 0**

515.  `void main()`

```
        {
        char *s="Hello World";
        printf("%c",s);
        }
```

**ans: garbage character**

516.  void main()
```
        {
        char *s="Hello World";
        printf("%c",*s);
        }
```

**ans: H**

517.  fp,fs;
```
      fp=fopen("tc.dat","w");
      fs=fopen("tc.dat","w");
      putch('A',fp);
      putch('B',fs); What will happen?
```

**ans: A is overwritten by B**

518.  What is the equivalent of a[i]
      **ans: *(a+i)**

519.  **int (*func)(int,int)** is a pointer to a function with 2
      integers as parameters and  returning an integer value.

520.  **int *(*func)(int *,int *)** is a pointer to a function with 2
      integer pointers as parameters and returning a pointer to an
      integer

521.  switch(float value)

         **ans: compiler error**

522.  main()
```
        {
        int a[5]={1,2,3,4,5};
        int *p=a+1;
        int *q=a+5;
        int dif=q-p;
        printf("%d", dif);
        }
```

         **ans: 4**

523.  switch(NULL)
         **ans: case 0: will be executed.**

```
524. #define exp 5
     main()
           {
           printf("%d",exp++);
           }

           ans: lvalue required

525. strcat(str,str);
           ans: compilation error (destination string length should
           accommodate both the strings)

526. int(*ptr)[10]

           ans: pointer to array of 10 integers.

527. int main()
           {
           char *str = "Hello, world" ;
           printf("%5s" , str);
           }

           ans: Hello, world (when the field width is less than the
           length of the string the entire string is printed)

528. int *ptr[10];

           ans: declaration of 10 pointers

529. int main()
           {
           extern int i;
           printf("%d" , i);
           }

           ans: linker error

530. void temp();
     void temp(void);
     int main()
           {
           temp();
           }
           void temp()
           {
           printf("C is exciting!");
           }

           ans: C is exciting!

531. void temp();
```

```
    void temp(void);
    int main()
        {
        temp();
        }
        void temp(void)
        {
        printf("C is exciting!");
        }

        ans: C is exciting!
```

532.  `void temp();`

```
    void temp(void);
    int main()
        {
        temp(void);
        }
        void temp()
        {
        printf("C is exciting!");
        }

        ans: compiler error (syntax error)
```

533.  `void temp(int i)`

```
        {
        if(i == 10) return;
        i++ ;
        temp(i);
        printf("%d " , i);
        }
    int main()
        {
        temp(1);
        }

        ans: 10 9 8 7 6 5 4 3 2
```

534.  some question on "strtok" function

535.  `int main()`

```
        {
        char *str = "Hello, world";
        int i = sizeof(str);
        for( ; i >= 0 ; i--)
        printf("%c" , str[i]);
        }

        ans: olleH (sizeof pointer is 4 bytes)
```

```
536.  int main()
         {
         int a = MAX( 4+2 , 3*2) ;
         printf(" %d " , a);
         }

         ans: 6

537.  main()
         {
         int x;
         printf("\n%d",x=0,x=20,x=40);
         }

         ans: 0

538.  main()
         {
         int a[]={1,2,5,6,9,10};
         int *b=&a[4];
         printf("\n%d",b[-3]);
         }

         ans: 2

539.  main()
         {
         int x=0,y=1;
         if(x=y)
         y= 7;
         else
         y=2;
         printf("%d", y);
         }

         ans: 7

540.  main()
         {
         int i=39,count=0;
         while( i & 1) //some condition like this
         {
         count++;
         i=i>>1;
         }
         printf("%d",count);
         }

         ans: 3

541.  main()
```

```
        {
        int i=39,count=0;
        while( i & 1) //some condition like this
        {
        count++;
        i>>1;
        }
        printf("%d",count);
        }

        ans: infinite loop
```

542.  main()
```
        {
        int x=128;
        printf("\n%d",1+x++);
        }
```

        **ans: 129**

543.  main()
```
        {
        FILE *f1;
        FILE *f2;
        f1=fopen("myfile","w");
        f2=fopen("myfile","w");
        fputc('A',f1);
        fputc('B',f2);
        fclose(f1);
        fclose(f2);
        }
    what does f1 n f2 conatins?
```

        **ans: B**

544.  if i/p is **code friday monday sunday** in commad line then
```
    main(int argc,char *argv[])
        {
        printf("\n%c",**++argv);
        }
```

        **ans:may be f**

545.  #define max 10
```
    main()
        {
        printf("\n%d",max++);
        }
```

        **ans: error (lvalue required)**

```
546.  main()
          {
          int a[]={1,2,9,8,6,3,5,7,8,9};
          int *p=a+1;
          int *q=a+6;
          printf("\n%d",q-p);
          }

          ans: 5

547.  main()
          {
          int i=3;
          while(i--){
          int i=100;
          i--;
          printf("%d ",i);
          }
          }

          ans: 99 99 99

548.  what does (*a)[10] means?

          ans: a is pointer to an array of 10 integers

549.  Open a file "input" and print the odd number of lines first on
the screen and then
      even number of lines..something like that.....

550.  main()
          {
          int x=5, y;
          y= x*x++ * ++x ;
          printf("%d %d",x,y);
          }

          ans: 7 216

551.  main()
          {
          int a=10,b=5;
          while(--b>=0 && ++a)
          {
          --b;
          ++a;
          }
          printf("%d %d",a,b);
          }
```

**ans: 16 -2**

552. ```c
main()
{
char i;
for (i=0; i<=255; i++)
{
printf("%c", i);
}
}
```

**ans: infinite loop ( signed char range is -128 to 127)**

553. ```c
main()
{
int i=0;
switch(i)
{
case 1: printf("hi");
case 0: printf("zero");
case 2: printf("world");
}
}
```

**ans: zeroworld**

554. ```c
struct XXX
{
int a:6;
float b:4;
char s;
}structure;
main()
{
printf("%d",sizeof(structure));
}
```

**ans: error (bit fields must be signed or unsigned int)**

555. ```c
struct XXX
{
int a:6;
/*float b:4;*/
char s;
}structure;
main()
{
printf("%d",sizeof(structure));
}
```

**ans: 2**

```
556.  struct XXX
            {
            int a:6;
            /*char s;*/
            }structure;
      main()
            {
            printf("%d",sizeof(structure));
            }

            ans: 1

557.  struct XXX
            {
            int a;
            char s;
            }structure;
      main()
            {
            printf("%d",sizeof(structure));
            }

            ans: 3

558.  main()
            {
            char *s;
            s="hot java";
            strcpy(s,"solaris java");
            printf("%s",s);
            }

            ans: solaris java (extra locations will be overwritten)

559.  main()
            {
            char *p='a';
            int *i=100/ *p;
            printf("%d",i);
            }

            ans: error (nonportable pointer conversion)

560.  main()
            {
            int n=5;
            printf("\nn=%*d",n,n);
            }

            ans: n=      5 (width specifier %5d right justified)
```

561. How long the following program will run?
```
main()
    {
    printf("\nSonata Software");
    main();
    }
```

**ans: until the stack overflows**

562. 
```
main()
    {
    const int x=5;
    int *ptrx;
    ptrx=&x;
    *ptrx=10;
    /*x=10;*/
    printf("%d",x);
    }
```

**ans: 10 (you can change a constant object by using a pointer)**

563. 
```
main()
    {
    const int x=5;
    int *ptrx;
    ptrx=&x;
    *ptrx=10;
    x=15;
    printf("%d",x);
    }
```

**ans: error (cannot modify a constant object)**

564. 
```
main()
    {
    const char *fun();
    *fun()="A";
    }
const char *fun()
    {
    return "Hello";
    }
```

**ans: error (cannot modify a constant object) fun() returns to a "const char" pointer which cannot be modified**

565. What error would the following function give on compilation?
```
f(int a, int b)
```

```
{
int a;
a=20;
return a;
}
```

**ans: error (redeclaration of a)**

566.  Would the following program compile?
```
main()
{
int a=10,*j;
void *k; j=k=&a;
j++;
k++;
printf("\n%u%u",j,k);
}
```

**ans: No, the arithmetic operation is not permitted on void pointers. Size of the type is unknown.**

567.  In the following program how would you print 50 using p?
```
main()
{
int a[]={10, 20, 30, 40, 50};
char *p;
p= (char*) a;
}
```

**ans: printf("%d",*((int*)p+4)); or printf("%d",*(p+8));**

568.  Point out the error in the following program
```
main()
{
int a=10;
void f();
a=f();
printf("\n%d",a);
}
void f()
{
printf("\nHi");
}
```

**ans: error (not an allowed type). The program is trying to collect the value of a "void" function into an integer variable.**

569.  If the following program (myprog) is run from the command line as **myprog friday tuesday sunday,** What would be the output?
```
main(int argc, char *argv[])
```

```
        {
        while(sizeof(argv))
        printf("%s",argv[--sizeof(argv)]);
        }
```

**ans:**

570. If the following program (myprog) is run from the command line
     as **myprog friday tuesday sunday,** What would be the output?
     main(int argc, char *argv[])
```
        {
        printf("%c",*++argv[1]);
        }
```

**ans: r (check it out)**

571. If the following program (myprog) is run from the command line
     as myprog friday tuesday sunday, What would be the output?
     main(int argc, char*argv[])
```
        {
        printf("%c",**++argv);
        }
```

**ans: f (check it out)**

572. main()
```
        {
        char near * near *ptr1;
        char near * far *ptr2;
        char near * huge *ptr3;
        printf("%d %d
%d",sizeof(ptr1),sizeof(ptr2),sizeof(ptr3));
        }
```

**ans: 2 4 4**

573. What is the difference between the following declarations?
     const char *const s; char const *const s;

**ans. No difference**

574. What is the difference between the following declarations?
     const char *s;
     char const *s;

**ans. No difference**

575. main()
```
        {
        int y=128;
        const int x; x=y;
```

```
            printf("%d",x);
            }
```

**ans: error (cannot modify a constant object)**

576.  main()
```
            {
            int y=128;
            const int x=y;
            printf("%d",x);
            }
```

**ans: 128 (when not initialized const variable will have garbage value)**

577.  main()
```
            {
            const int x;
            x=128;
            printf("%d",x);
            }
```

**ans: error (cannot modify a constant object. x should have been initialized where it is declared)**

578.  In the following code, is p2 an integer or an integer pointer?
      **typedef int* ptr**
      **ptr p1,p2;**

      **ans. Integer pointer**

579.  If the following program (myprog) is run from the command line
      as **myprog monday tuesday wednesday thursday**, What would be the
      output?
      main(int argc, char *argv[])
```
            {
            while(--argc >0)
            printf("%s",*++argv);
            }
```

      **ans: monday tuesday wednesday Thursday**

580.  If the following program (myprog) is run from the command line
      as myprog 1 2 3, What would be the output?
      main(int argc, char *argv[])
```
            {
            int i,j=0;
            for(i=0;i<argc;i++)
            j=j+ atoi(argv[i]);
            printf("%d",j);
            }
```

**ans: check out**

581.  If the program (myprog) is run from the command line as myprog
      1 2 3 , What would be the output?
      main(int argc, char *argv[])
```
            {
            int i;
            for(i=0;i<argc;i++)
            printf("%s",argv[i]);
            }
```

      **ans: C:\MYPROG.EXE 1 2 3**

582.  main()
```
            {
            FILE *fp;
            fp= fopen("trial","r");
            }
            fp points to:
```

      **ans: A structure which contains a "char" pointer which
      points to the first character in the file.**

583.  What is the type of the variable b in the following
declaration?
      #define FLOATPTR float*
      FLOATPTR a,b;

      **ans: float**

584.  **#define FLOATPTR float***
      main()
```
            {
            FLOATPTR a,b;
            b=10.0;
            }
```

      **ans: b is a float variable (no error)**

585.  **typedef float* FLOATPTR;**
      main()
```
            {
            FLOATPTR a,b;
            b=10.0;
            }
```

      **ans: error (illegal use of floating point. Here b is a
      floating pointer variable. Observe the difference
      between marco and typedef in 584 and 585 problems)**

```
586.  #define SQR(x) (x*x)
      main()
            {
            int a,b=3;
            a= SQR(b+2);
            printf("%d",a);
            }
```

**ans: 11**

```
587.  main()
            {
            int i=4;
            switch(i)
            {
            default:
            printf("\n A mouse is an elephant built by the
Japanese");
            case 1:
            printf(" Breeding rabbits is a hair raising
experience");
            break;
            case 2:
            printf("\n Friction is a drag");
            break;
            case 3:
            printf("\n If practice make perfect, then nobody's
perfect");
            }
            }
```

**ans: A mouse is an elephant built by the Japanese**
**Breeding rabbits is a hair raising experience**

588.  In the following code, in which order the functions would be
called?

```
      a= f1(23,14)*f2(12/4)+f3();
```

**ans: f1, f2, f3**

```
589. f3()
            {
            printf("three ");
            return 1;
            }

      f1(int x, int y)
            {
            printf("one ");
            return(x+y);
            }
```

```
f2(int x)
      {
      printf("two ");
      return x;
      }

main()
      {
      int a;
      a= f1(23,14)*f2(12/4)+f3();
      printf("%d",a);
      }

      ans: one two three 112
```

590. 
```
main()
      {
      int a=10,b;
      a<= 5 ? b=100 : b=200;
      printf("\n%d",b);
      }

      ans: error (lvalue required. Conditional operator has
      highest priority than assignment operator)
```

591. 
```
main()
      {
      int a=10,b;
      a<= 5 ? b=100 : (b=200);
      printf("\n%d",b);
      }

      ans: 200
```

592. 
```
main()
      {
      int a=10,b;
      a>= 5 ? b=100 : (b=200);
      printf("\n%d",b);
      }

      ans: 100
```

593. 
```
main()
      {
      int i=1;
      switch(i)
      {
      case 1:
      printf("\nRadioactive cats have 18 half-lives");
      break;
```

```
            case 1*2+4:
            printf("\nBottle for rent -inquire within");
            break;
            }
            }
```

**ans:  Radioactive cats have 18 half-lives (no error)**

594.  main()
```
            {
            int i=2;
            printf("I=%d i=%d",++i,++i);
            }
```

**ans: I=4 i=3**

595.  main()
```
            {
            unsigned char i=0x80;
            printf("i=%d",i<<1);
            }
```

**ans: i=256**

596.  main()
```
            {
            unsigned char i=0x80;
            i=i<<1;
            printf("i=%d",i);
            }
```

**ans: i=0**

597.  main()
```
            {
            int B=0xFFFF;
```
            **~B ;**                                  /*  **note: not assigned
to B */**
```
            printf("%d",B);
            }
```

**ans: -1**

598.  main()
```
            {
            unsigned int B=0xFFFF;
            ~B ;
```
            **printf("%d",B);**
```
            }
```

**ans: -1**

599.  main()
```
        {
        unsigned int B=0xFFFF;
        ~B ;
        printf("%u",B);
        }
```

**ans: 65535**

600.  Func(int a, intb)
```
        {
        int a;
        a=10;
        return a;
        }
        will there be any error?
```

**ans: error (redeclaration of a)**

601.  string is given **myprog one two three** Where myprog is an exe
      file. What will the output of the following program ?

```
main(int argc, char *argv[])
        {
        printf("%c"++**argv);
        }
```

**ans: n (check it out)**

602.  **#define SQR(b) b*b;**
```
main()
        {
        int i=3;
        printf("%d",SQR(i+2));
        }
```

**ans: error (semicolon in macro definition will cause
error when it is replaced in printf statement)**

603.  #define SQR(b) b*b
```
main()
        {
        int i=3;
        printf("%d",SQR(i+2));
        }
```

**ans: 11**

604.  main()
```
        {
```

```
       char c='a';
       printf("%d %d", sizeof(c),sizeof('a'));
       }
```

**ans: 1 2**

605.  main()
```
       {
       char c='a';
       Printf("%d %d", sizeof(c),sizeof('a'));
       }
```

**ans: linker error (undefined symbol_Printf)**

606.  main()
```
       {
       Char c='a';
       printf("%d %d", sizeof(c),sizeof('a'));
       }
```

**ans: error (undefined symbol 'Char' , undefined symbol 'c' , statement missing ; )**

607.  void main(void)
```
       {
       struct s
       {
       int x;
       float y;
       }s1={25,45.00};
       union u
       {
       int x;
       float y;
       }u1;
       u1=(union u)s1;
       printf("%d and %f",u1.x,u1.y);
       }
```

**ans: error (incompatible type conversion)**

608.  int fn(void);
      void print(int,int(*)());
```
       int i=10;
```

      void main(void)
```
       {
       int i=20;
       print(i,fn);
       }
```

```
void print(int i,int (*fn1)())
     {
     printf("%d\n",(*fn1)());
     }

int fn(void)
     {
     return(i-=5);
     }

     ans: 5
```

609. 
```
void main(void)
     {
     char numbers[5][6]={"Zero","One","Two","Three","Four"};
     printf("%s is %c",&numbers[4][0],numbers[0][0]);
     }

     ans: Four is Z
```

610. 
```
void main(void)
     {
     int y,z;
     int x=y=z=10;
     int f=x;
     float ans=0.0;
     f *=x*y;
     ans=x/3.0+y/3;
     printf("%d %.2f",f,ans);
     }

     ans: 1000 6.33
```

611. 
```
double dbl=20.4530,d=4.5710,dblvar3;
void main(void)
     {
     double dbln(void);
     dblvar3=dbln();
     printf("%.2f\t%.2f\t%.2f\n",dbl,d,dblvar3);
     }
double dbln(void)
     {
     double dblvar3;
     dbl=dblvar3=4.5;
     return(dbl+d+dblvar3);
     }

     ans: 4.50      4.57     13.57
```

612. 
```
void main(void)
     {
```

```
        int oldvar=25,newvar=-25;
        int swap(int,int);
        swap(oldvar,newvar);
        printf("Numbers are %d\t%d",newvar,oldvar);
        }
     int swap(int oldval,int newval)
        {
        int tempval=oldval;
        oldval=newval;
        newval=tempval;
        }

        ans: Numbers are -25 25


613.  void main(void)
        {
        int i=100,j=20;
        i++ =j;
        i*=j;
        printf("%d\t%d\n",i,j);
        }

        ans: error (lvalue required)


614.  int newval(int);
     void main(void)
        {
        int ia[]={12,24,45,0};
        int i;
        int sum=0;
        for(i=0;ia[i];i++)
        {
        sum+=newval(ia[i]);
        }
        printf("Sum= %d",sum);
        }
     int newval(int x)
        {
        static int div=1;
        return(x/div++);
        }

        ans: Sum= 39


615.  void main(void)
        {
        int var1,var2,var3,minmax;
        var1=5;
        var2=5;
        var3=6;
```

```
          minmax=(var1>var2)?(var1>var3)?var1:var3:(var2>var3)?
var2:var3;
          printf("%d\n",minmax);
          }

          ans: 6 (maximum of three numbers)


616.  static int i=50;
      int print(int i);
      void main(void)
          {
          static int i=100;
          while(print(i))
          {
          printf("%d ",i);
          i--;
          }
          }
      int print(int x)
          {
          static int i=2;
          return(i--);
          }

          ans: 100 99


617.  void main(void);
      typedef struct NType
          {
          int i;
          char c;
          long x;
          }NewType;

      void main(void)
          {
          NewType *c;
          c=(NewType *)malloc(sizeof(NewType));
          c->i=100;
          c->c='C';
          (*c).x=100L;
          printf("(%d,%c,%4Ld)",c->i,c->c,c->x);
          }

          ans: (100,C, 100)


618.  main()
          {
          char *p1="Name";
          char *p2;
          p2=(char *)malloc(20);
```

```
        while(*p2++=*p1++);
        printf("%s\n",p2);
        }
```

**ans: an empty string (no output)**

619.  main()
```
        {
        int x=20,y=35;
        x = y++ + x++;
        y = ++y + ++x;
        printf("%d %d\n",x,y);
        }
```

**ans: 57 94**

620.  main()
```
        {
        int x=5;
        printf("%d %d %d\n",x,x<<2,x>>2);
        }
```

**ans: 5 20 1**

621.  #define swap1(a,b) a=a+b;b=a-b;a=a-b;
      main()
```
        {
        int x=5,y=10;
        swap1(x,y);
        printf("%d %d\n",x,y);
        swap2(x,y);
        printf("%d %d\n",x,y);
        }
      int swap2(int a,int b)
        {
        int temp;
        temp=a;
        b=a;
        a=temp;
        return;
        }
```

**ans: 10 5**
**    10 5**

622.  #define swap1(a,b) a=a+b;b=a-b;a=a-b;
      main()
```
        {
        int x=5,y=10;
        swap1(x,y)
        printf("%d %d\n",x,y);
```

```
        swap2(x,y);
        printf("%d %d\n",x,y);
        }
int swap2(int a,int b)
        {
        int temp;
        temp=a;
        b=a;
        a=temp;
        return;
        }

        ans: 10 5
             10 5
```

623.  #define swap1(a,b) a=a+b;b=a-b;a=a-b
      main()
```
        {
        int x=5,y=10;
        swap1(x,y)
        printf("%d %d\n",x,y);
        swap2(x,y);
        printf("%d %d\n",x,y);
        }
int swap2(int a,int b)
        {
        int temp;
        temp=a;
        b=a;
        a=temp;
        return;
        }

        ans: error (statement missing ;)
```

624.  main()
```
        {
        char *ptr = "Ramco Systems";
        (*ptr)++;
        printf("%s\n",ptr);
        ptr++;
        printf("%s\n",ptr);
        }

        ans: Samco Systems
              amco Systems
```

625.  main()
```
        {
        char s1[]="Ramco";
        char s2[]="Systems";
```

```
        s1=s2;
        printf("%s",s1);
        }
```

**ans: error (lvalue required)**

626. main()
```
        {
        char *p1;
        char *p2;
        p1=(char *) malloc(25);
        p2=(char *) malloc(25);
        strcpy(p1,"Ramco");
        strcpy(p2,"Systems");
        strcat(p1,p2);
        printf("%s",p1);
        }
```

**ans: RamcoSystems**

627. main()
```
        {
        int x=10,y=15;
        x=x++;
        y=++y;
        printf("%d %d\n",x,y);
        }
```

**ans: 11 16**

628. main()
```
        {
        int a=0;
        if(a=0) printf("Ramco Systems\n");
        printf("Ramco Systems\n");
        }
```

**ans: Ramco Systems**

629. main()
```
        {
        int a=0;
        if(a==0) printf("Ramco Systems\n");
        printf("Ramco Systems\n");
        }
```

**ans: Ramco Systems**
         **Ramco Systems**

630. int SumElement(int *,int);
     void main(void)
```

```
          {
          int x[10];
          int i=10;
          for(;i;)
          {
          i--;
          *(x+i)=i;
          }
          printf("%d",SumElement(x,10));
          }
     int SumElement(int array[],int size)
          {
          int i=0;
          float sum=0;
          for(;i<size;i++)
          sum+=array[i];
          return sum;
          }
```

          **ans: 45**

631.  `int printf(const char*,...);`
      `void main(void)`
```
          {
          int i=100,j=10,k=20;
          int sum;
          float ave;
          char myformat[]="ave=%.2f";
          sum=i+j+k;
          ave=sum/3.0;
          printf(myformat,ave);
          }
```

          **ans: ave=43.33**

632.  `void main(void)`
```
          {
          int a[10];
          printf("%d",((a+9) + (a+1)));
          }
```

          **ans: error (invalid pointer addition)**

633.  `int bags[5]={20,5,20,3,20};`
      `void main(void)`
```
          {
          int pos=5,*next();
          *next()=pos;
          printf("%d %d %d",pos,*next(),bags[0]);
          }
```
      `int *next()`

```
        {
        int i;
        for(i=0;i<5;i++)
        if (bags[i]==20)
        return(bags+i);
        printf("Error!");
        exit(0);
        }

        ans: 5 20 5
```

634.  ```
      static int i=5;
      void main(void)
          {
          int sum=0;
          do
          {
          sum+=(1/i);
          }while(0<i--);
          }
      ```

      **ans: error (divide by zero exception)**

635.  ```
      void main(void)
          {
          void pa(int *a,int n);
          int arr[5]={5,4,3,2,1};
          pa(arr,5);
          }
      void pa(int *a,int n)
          {
          int i;
          for(i=0;i<n;i++)
          printf("%d ",*(a++)+i);
          }
      ```

      **ans: 5 5 5 5 5**

636.  ```
      const int k=100;
      void main(void)
          {
          int a[100];
          int sum=0;
          for(k=0;k<100;k++)
          *(a+k)=k;
          sum+=a[--k];
          printf("%d",sum);
          }
      ```

      **ans: error (cannot modify a constant object)**

```
637.  int k=100;
      void main(void)
            {
            int a[100];
            int sum=0;
            for(k=0;k<100;k++)
            *(a+k)=k;
            sum+=a[--k];
            printf("%d",sum);
            }

            ans: 99

638.  main()
            {
            printf("Hello %d",printf("QUARK test? "));
            }

            ans: QUARK test? Hello 12

639.  main()
            {
            int i,j,A;
            for (A = -1;A<=1; A++)
            printf("%d ",!!A);
            }

            ans: 1 0 1

640.  main()
            {
            int i=255;
            printf("%d\t",++(i++));
            }

            ans: error (lvalue required)

641.  main()
            {
            char i = 'a';
            printf("%c %c",i,(++i));
            }

            ans: b b

642.  main()
            {
            int i,j;
            printf("QUARK %s\n",main());
            }
```

```
        ans: There is nothing on the screen and prog waits till
        the memory lasts and then out of memory run time error.
```

643.  #define f(x) x*x*x
      main()
          {
          printf("\n%d",f(2+2));
          }

          ans: 12

644.  main()
          {
          void fun1(void *);
          char a[] = "quark";
          void *temp;
          temp = a;
          fun1(temp);}
      void fun1(void *temp1 )
          {
          int t1 = 0;
          while(*((char*)temp1+ t1++ )!='\0') {
          printf("%c",*((char*)temp1 + t1));
          }
          }

          ans: uark

645.  void main()
          {
          int x=3;
          printf("%d %d",x>>1, x<<3);
          }

          ans: 1 24

646.  void main()
          {
          int *x;
          x =(int *) 15;
          }

          ans: Location 15 in the program space is assigned to
pointer x

647.  Which of the following functions cannot be called from another
file?

      a. const void func(){ ……..}
      b. extern void func(){………}
      c. void func(){………}

d. static void func(){.........}

      **ans. static**

648. int *func()
```
        {
        static int x=0;
        x++; return &x;
        }
int main()
        {
        int * y = func();
        printf("%d ",(*y)++);
        func();
        printf("%d",*y);
        return 0;
        }
```

      **ans: 1 3**

649. void main()
```
        {
        unsigned int x= -1;
        int y =0;
        if(y<=x) printf("A is true\n");
        if (y ==(x = -10)) printf("B is true\n");
        if ((int) x>=y) printf("C is true\n");
        }
```

      **ans: A is true**

650. void main()
```
        {
        int x= -1;
        int y =0;
        if(y<=x) printf("A is true\n");
        if (y ==(x = -10)) printf("B is true\n");
        if ((int) x>=y) printf("C is true\n");
        }
```

      **ans: no output**

651. void main()
```
        {
        unsigned int x= -1;
        int y =0;
        printf("%d ",x);
        if(y<=x) printf("A is true\n");
        if (y ==(x = -10)) printf("B is true\n");
        if ((int) x>=y) printf("C is true\n");
        }
```

652. void main()

```
        {
        unsigned int x= -1;
        int y =0;
        printf("%u ",x);
        if(y<=x) printf("A is true\n");
        if (y ==(x = -10)) printf("B is true\n");
        if ((int) x>=y) printf("C is true\n");
        }
```

        **ans: 65535 A is true (%u unsigned integer specifier)**

653. In the following code what is the correct way to increment the variable ptr to
    point to the next member of the array

```
    union intfloat
        {
        int intArray[ 5];
        float floatArray[ 5];
        };
    union intfloat arr[20];
    void *ptr =arr;
```

        **ans: ptr = (void*)((union intfloat*)ptr +1);**

654. #define PRINTXYZ(x,y,z) printf (#x "=%d\t" #z "=%d\n", x, y)

```
    void main()
        {
        int x, y, z;
        x=0; y=1; z=2;

        x ||  ++y ||++z;
        PRINTXYZ(x,y,z);

        ++x ||  ++y && ++z;
        PRINTXYZ(x,y,z);

        ++x && ++y ||  ++z;
        PRINTXYZ(x,y,z);
        }
```

        **ans:**

           **x=0    z=2**
           **x=1    z=2**
           **x=2    z=3**

655. main()

```
{
printf("%d %d", sizeof(NULL), sizeof(""));
}
```

**ans: 4 1 (NULL is a pointer so it takes 4 bytes. sizeof empty string is 1)**

656.
```
int *check(int,int);
void main()
    {
    int c,d;
    c = check(11,29);
    d= check(20,30);
    printf("\nc=%u",c);
    }
int * check(int i,int j )
    {
    int *p, *q;
    p=&i;
    q=&j;
    if(i>=95)
    return(q);
    else
    return(p);
    }
```

**ans: nonportable pointer conversion**

657.
```
void main()
    {
    int a[3][2]={ 1,8,5,7,6,8};
    printf("%d",((a+1)-(&a+1)));
    }
```

**ans: −2. I haven't been able to figure this one out. a is the address of the 2-d array, here a, &a, *a all give the same value, i.e., address of the array. (a+1) gives the address of the second row, it is the same as a[1]. *(a+1) gives the address of the first cell of the second row. **(a+1) gives the value of the element stored in the first cell in the second row. (*(a+1)+1) gives the address of the second cell of the second row. *(*(a+1)+1) gives the value of the element stored in the second cell in the second row.**

658.
```
void main()
    {
    int a[3][2]={ 1,8,5,7,6,8};
    printf("%d ",a);
    printf("%d ",&a);
    printf("%d ",*a);
```

```
            }

            ans: 8682 8682 8682 (all are same)


659.  main()
            {
            char str1[]="Hello";
            char str2[]="Hello";
            if(str1==str2 && (*(str1+6)== *(str2+6)) )
            printf("\n Equal");
            else
            printf("\n unequal");
            }

            ans: unequal


660.  main()
            {
            int a, b=255,c=127;
            a=~b;
            c=c^(~a & b|0);
            c=c^(~(~b));
            printf("%d\n",c);
            }

            ans: 127


661.  #define f(a,b) a+b
      #define g(x,y) x*y
      main()
            {
            int i;
            i=f(4,g(5,6));
            printf("%d",i);
            }

            ans: 34


662.  main()
            {
            int i,j=9999;
            char buff[5];
            i=sprintf(buff,"%d",j);
            printf("%d %s",i,buff);
            }

            ans: 4 9999


663.  main()
            {
            int i,j=99999;
```

```
        char buff[5];
        i=sprintf(buff,"%d",j);
        printf("%d %s",i,buff);
        }
```

**ans: 6 –31073**

664.  main()
```
        {
        int I=2;
        int j=3;
        int k=4;
        printf("%d",(I<j<k));
        }
```

**ans: 1**

665.  #define macro(a) ((a++) + (++a) + (a++))
      main()
```
        {
        printf("%d",macro(1));
        }
```

**ans: error (lvalue required)**

666.  int func(int I)
```
        {
        static int k=0;
        k++;
        if(k>5)
        return 1;
        else
        return func(I-1);
        }
```

      int main()
```
        {
        printf("%d",func(1));
        }
```

**ans: 1**

667.  main()
```
        {
        char *str="quark" "media";
        printf("%s",str);
        }
```

**ans: quarkmedia**

668.  main()
```
```

```
{
char *str;
str="hello"    "india";
printf("%s",str);
}
```

**ans: helloindia**

669. main()
```
{
int i=0,z;
z=sizeof(++i + i++);
printf("%d %d",z,i);
}
```

**ans: 2 0 (the operand of a sizeof operator is either an
expression, which is not evaluated, or a parenthesized
type name)**

670. main()
```
{
int y=10;
for (int x=0;x<=10;x++);
y+=x;
printf("%d",y);
}
```

**ans: error (x should be declared before for loop)**

671. main()
```
{
int y=10,x;
for (x=0;x<=10;x++);
y+=x;
printf("%d",y);
}
```

**ans: 21**

672. fun(int a)
```
{
static int b;
}
```
what is the storage allocation for both a and b?

**ans: a-stack, b-bss (block starting with symbol)**

673. int *fun(int a)
```
{
return (&a);
}
```

```
int *fun(int a)
    {
    int *b;
    b=&a;
    return(b);
    }
int *fun(int a )
    {
    int *b;
    b=malloc(sizeof(int));
    b=&a;
    return (b);
    }
which of the following functions are not correct?
```

**ans: 1 & 2 are not correct**

674.  ```
      int fun(int a,int y)
          {
          int x;
          x=a+y;
          return (x);
          }
      int main()
          {
          int x,y=1,z=0,c;
          z=fun(y,c);
          printf(" %d ",x);
          }
      ```

**ans: garbage value**

675.  ```
      main()
          {
          int i;
          printf("%d",++i++);
          }
      ```

**ans: error (lvalue required)**

676.  ```
      main()
          {
          int a=2;
          printf("%d %d %d",++a,a++);
          }
      ```

**ans: 4 2 garbage value**

677.  ```
      struct abc
          {
          char a[10];
      ```

```
        int a,b;
        };
main()
        {
        struct abc ab={"main"};
        printf("%d %d",ab.a,ab.c);
        }
```

        **ans: error (multiple declaration of a and undefined
symbol c)**

678.  void main()
```
        {
        printf("persistent");
        main();
        }
```

        **ans: till stack overflows**

679.  func(char *s1,char * s2)
```
        {
        char *t;
        t=s1;
        s1=s2;
        s2=t;
        }
void main()
        {
        char *s1="jack", *s2="jill";
        func(s1,s2);
        printf("%s %s",s1,s2);
        }
```

        **ans: jack jill**

680.  func(char *s1,char * s2)
```
        {
        char *t;
        printf("%s %s ",s1,s2);
        t=s1;
        s1=s2;
        s2=t;
        printf("%s %s ",s1,s2);
        }
void main()
        {
        char *s1="jack", *s2="jill";
        func(s1,s2);
        printf("%s %s",s1,s2);
        }
```

681. void main()
```
     {
     int a[5] ={1,2,3,4,5},i,j=2;
     for (i =0;i<5;i++ )
     func(j,&a[i]);
     for (i =0;i<5;i++ )
     printf("%d ",a[i]);
     }
func(int j,int *a)
     {
     j=j+1;
     a=a+j;
     }
```

**ans: 1 2 3 4 5**

682. void main()
```
     {
     int a[5] ={1,2,3,4,5},i,j=2;
     for (i =0;i<5;i++ )
     func(j,a[i]);
     for (i =0;i<5;i++ )
     printf("%d ",a[i]);
     }
func(int j,int *a)
     {
     j=j+1;
     a=a+j;
     }
```

**ans: 1 2 3 4 5**

683. main()
```
     {
     for (a=1;a<=100;a++)
     for(b=a;b<=100;b++)
     foo();
     }
     foo()
     {}
     how many times foo will be called?
```

**ans: 5050**

684. int i;
```
main()
     {
     int a,b;
     for (a=1;a<=100;a++)
```

```
        for(b=a;b<=100;b++)
        foo();
        printf("%d",i);
        }
    foo()
        {
        i++;
        }
```

**ans: 5050**

685.  One palindrome programme was given in recursion

   **ans : pal(f++,t--)**

686.  main()
```
        {
        int i=foo(2);
        printf("%d",i);
        }
    foo(int s)
        {
        if(!s)
        return s;
        else
        {
        int i=5;
        return i;
        }
        }
```

   **ans: 5**

687.  main()
```
        {
        int k=0,i=0,j=1;
        if(!0&&(k=2)) printf("%d ",k);
        if(!0||(k=0))
        printf("%d",k);
        }
```

   **ans: 2 2**

688.  main()
```
        {
        int k=0,i=0,j=1;
        if(!0&&k=2) printf("%d ",k);
        if(!0||k=0)
        printf("%d",k);
        }
```

689.  main()
```
        {
        int i;
        for(i=0;i<3;i++)
        switch(i)
        {
        case 1: printf("%d",i);
        case 2 : printf("%d",i);
        default: printf("%d",i);
        }
        }
```

        **ans: 011122**

690.  **int *num={10,1,5,22,90};**
      main()
```
        {
        int *p,*q;
        int i;
        p=num;
        q=num+2;
        i=*p++;
        printf("%d %d",i,q-p);
        }
```

        **ans: error (declaration error)**

691.  **int num[]={10,1,5,22,90};**
      main()
```
        {
        int *p,*q;
        int i;
        p=num;
        q=num+2;
        i=*p++;
        printf("%d %d",i,q-p);
        }
```

        **ans: 10 1**

692.  **int *(*p[10])(char *, char*)**

        **ans: array of pointers to function with two character
        pointers as arguments and returning interger pointer**

693.  main()
```
        {
        char *a[4]={"jaya","mahe","chandra","buchi"};
```

```
        printf("%d %d %d",sizeof(a),sizeof(char
*),sizeof(a)/sizeof(char *));
        }
```

**ans: 16 4 4**

694.  void fn(int *a, int *b)
```
        {
        int *t;
        t=a;
        a=b;
        b=t;
        }

main()
        {int a=2;
        int b=3;
        fn(&a,&b);
        printf("%d,%d", a,b);
        }
```

**ans: 2,3**

695.  #define scanf "%s is a string"
```
main()
        {
        printf(scanf,scanf);
        }
```

**ans: %s is a string is a string**

696.  main()
```
        {
        char *p="abc";
        char *q="abc123";
        while(*p=*q)
        printf("%c%c",*p,*q);
        }
```

**ans: prints a infinite times**

697.  main()
```
        {
        printf("%u",-1);
        }
```

**ans: 65535**

698.  #define void int
```
int i=300;
void main(void)
```

```
      {
      int i=200;
      {
      int i=100;
      printf("%d ",i);
      }
      printf("%d",i);
      }
```

**ans: error (parameter 1 missing name)**

699.  #define void int
      int i=300;
      void main(**void argc**)
```
      {
      int i=200;
      {
      int i=100;
      printf("%d ",i);
      }
      printf("%d",i);
      }
```

**ans: 100 200**

700.  main()
```
      {
      int x=2;
      x<<2;
      printf("%d ",x);
      }
```

**ans: 2**

701.  main()
```
      {
      int x=2;
      x=x<<2;
      printf("%d ",x);
      }
```

**ans: 8**

702.  main()
```
      {
      int a[]={0,0X4,4,9};
      int i=2;
      printf("%d %d",a[i],i[a]);
      }
```

**ans: 4 4**

```
703.  main()
          {
          int i=2+3,4>3,2;
          printf("%d",i);
          }

          ans: error

704.  main()
          {
          int i=(2+3,4>3,2);
          printf("%d",i);
          }

          ans: 2

705.  main()
          {
          int a=0,b=0;
          if(!a)
          {
          b=!a;
          if(b)
          a=!b;
          }
          printf("%d %d",a,b);
          }

          ans: 0 1

706.  main()
          {
          int I=10;
          I=I++ + ++I;
          printf("%d",I);
          }

          ans: 23

707.  swap(int x,y)
          {
          int temp;
          temp=x;
          x=y;
          y=temp;
          }

      main()
          {
          int x=2,y=3;
```

```
swap(x,y);
printf("%d %d",x,y);
}
```

**ans: error (swap function formal arguments declaration)**

708. **swap(int x, int y)**
```
{
int temp;
temp=x;
x=y;
y=temp;
}
```

```
main()
{
int x=2,y=3;
swap(x,y);
printf("%d %d",x,y);
}
```

**ans: 2 3**

709. struct
```
{
int x;
int y;
}abc;
```

x cannot be accessed by the following

```
1)abc-->x;
2)abc[0]-->x;
3)abc.x;
4)(abc)-->x;
```

**ans: 1 2 &4**

710. Automatic variables are destroyed after fn. ends because

a) Stored in swap
b) Stored in stack and poped out after fn. returns
c) Stored in data area
d) Stored in disk

**ans: b**

711. main()
```
{
int x=2,y=6,z=6;
x=y==z;
```

```
            printf("%d",x);
            }

            ans: 1
```

712.  i ) int *F()
      ii) int (*F)()

       **ans: The first declaraion is a function returning a
      pointer to an integer and the second is a pointer to a
      function returning int.**

713.  #define dprintf(expr) printf(#expr "=%d\n",expr)
      main()
```
            {
            int x=7;
            int y=3;
            dprintf(x/y);
            }
```

       **ans: x/y=2**

714.  main()
```
            {
            int i;
            char *p;
            i=0X89;
            p=(char *)i;
            p++;
            printf("%x %x\n",i,p);
            }
```

       **ans: 89 8a**

715.  main()
```
            {
            int i;
            char *p;
            i=0X89;
            p=(char *)i;
            p++;
            printf("%x %x\n",p,i);
            }
```

       **ans: 8a 0**

716.  The type of the controlling expression of a switch
      statement cannot be of the type

      a) int b) char c) short d)float e) none

**ans: d)float**

717. 
```c
main()
{
int X,b;
b=7;
X = b>8 ? b <<3 : b>4 ? b>>1:b;
printf("%d",X);
}
```

**ans: 3**

718. 
```c
main()
{
int n=2;
printf("%d %d\n", ++n, n*n);
}
```

**ans: 3 4**

719. 
```c
int x= 0x65;
main()
{
char x;
printf("%d\n",x);
}
```

**ans: unknown**

720. 
```c
main()
{
int a=10;
int b=6;
if(a=3)
b++;
printf("%d %d\n",a,b++);
}
```

**ans: 3 7**

721. 
```c
main()
{
enum Months {JAN =1,FEB,MAR,APR};
Months X = JAN;
if(X==1)
{
printf("Jan is the first month");
}
}
```

**ans: error**

```
722.  main()
         {
         enum Months {JAN =1,FEB,MAR,APR};
         enum Months X = JAN;
         if(X==1)
         {
         printf("Jan is the first month");
         }
         }

         ans: Jan is the first month

723.  main()
         {
         int l=6;
         switch(l)
         {
         default : l+=2;
         case 4: l=4;
         case 5: l++;
         break;
         }
         printf("%d",l);
         }

         ans: 5

724.  main()
         {
         int x=20;
         int y=10;
         swap(x,y);
         printf("%d %d",y,x+2);
         }
      swap(int x,int y)
         {
         int temp;
         temp =x;
         x=y;
         y=temp;
         }

         ans: 10 22

725.  #define INC(X) X++
      main()
         {
         int X=4;
         printf("%d",INC(X++));
         }
```

**ans: error (lvalue required)**

726.  main()
```
{
char s[]="Hello, world";
printf("%15.10s",s);
}
```

   **ans:       Hello, wor**

727.  main()
```
{
printf("%d\n",f(7));
}
f(x)
{
if(x<=4)
return x;
return f(--x);
}
```

   **ans: 4**

728.  main()
```
{
int x=0 ,*p=0;
x++;p++;
printf("%d and %d\n",p);
}
```

   **ans: 2 and 0**

729.  main()
```
{
int i=20,*j=&i;
f1(j);
*j+=10;
f2(j);
printf("%d and %d",i,*j);
}
f1(k)
int *k;
{ *k+=15;}
f2(x)
int *x;
{ int m=*x, *n=&m;
*n+=10;
}
```

   **ans: 45 and 45**

```
730.  func(int x)
           {
           if(x<=0)
           return (1);
           return func(x-1)+x;
           }
      main()
           {
           printf("%d",func(5));
           }

           ans: 16

731.  void funca(int *k)
           {
           *k+=20;
           }
      void funcb(int *k)
           {
           int m=*k,*n=&m;
           *n+=10;
           }
      main()
           {
           int var=25;
           int *varp=&var;
           funca(varp);
           *varp+=10;
           funcb(varp);
           printf("%d %d",var,*varp);
           }

           ans: 55 55

732.  main()
           {
           int x=0,*p=0;
           x++; p++;
           printf ("%d and %d\n",x,p);
           }

           ans: 1 and 2

733.  main()
           {
           int Y=10;
           if( Y++>9 && Y++!=10 && Y++>10)
           printf("%d",Y);
           else
           printf("........");
```

```
            }

            ans: 13

734.
      int i=10;
      main()
            {
            int i=20,n;
            for(n=0;n<=i;n++)
            {
            int i=10;
            i++;
            }
            printf("%d", i);
            }

            ans: 20

735.  main()
            {
            int i=20,j,k=0;
            for(j=1;j<i;j=1+4*(i/j))
            {
            k+=j<10?4:3;
            }
            printf("%d", k);
            }

            ans: 4

736.  main()
            {
            int i=10;
            printf("%d %d %d",i++,i++,i--);
            }

            ans: 10 9 10

737.  main()
            {
            int i=10;
            if(1,i++,++i)
            printf("The value for i is %d",i);
            }

            ans: The value for i is 12

738.  main()
            {
            int a=10,b=33;
```

```
                a=a^b;
                b=a^b;
                a=a^b;
                printf("%d %d", a,b);
                }

        ans: 33 10

739.  main()
                {
                int *a;
                int (*b)();
                printf("%d %d",sizeof(a),sizeof(b));
                }

        ans: 4 4

740.  main()
                {
                int i;
                char *p;
                i=0X89;
                p=(char *)i;
                p++;
                printf("%x\n",p);
                }

        ans: 8a

741.  main()
                {
                int x=0,*p=0;
                x++; p++;
                printf ("%d and %d\n",x,p);
                }

        ans: 1 and 2

742.  #define val 1+2
      main()
                {
                printf("%d %d",val/val,val^3);
                }

        ans: 5 0

743.  #define "this" "#"
      #define (x,y) x##y
      main()
                {
                printf("this","this is");
```

```
            }

            ans: error (define directive needs an identifier)

744.  main()
            {
            int a ,b=7;
            a=b<4?b<<1:b=4?71:a;
            printf("%d",a);
            }

            ans: error (lvalue required)

745.  main()
            {
            int a ,b=7;
            a=b<4?b<<1:(b=4?71:a);
            printf("%d",a);
            }

            ans: 71

746.  main()
            {
            int a,b;
            a=(10.15);
            b=10,15;
            printf("%d %d",a,b);
            }

            ans: 10 10 ('a' value is truncated, no effect of comma
            operator, it is just assignment)

747.  main()
            {
            int a,b;
            a=(10.15);
            b=(10,15);
            printf("%d %d",a,b);
            }

            ans: 10 15 ('a' value is truncated and effect of comma
            operator)

748.  main()
            {
            int a,b;
            a=(10,15);
            b=10,15;
            printf("%d %d",a,b);
            }
```

**ans: 15 10**

749.  #define VALUE 1+2
      main()
            {
            printf("%d and %d\n",VALUE/VALUE,VALUE*3);
            }

            **ans: 5 and 7**

750.  which of the following is not basic data type

            **ans: char * (pointers derived data types)**

751.  the declaration of the variable does not result in one of
      the following

            **ans: allocation of the storage space for the varable.**

752.  2 variables cannot have the same name if they are

            **ans: in the same block.**

753.  Which of the following is the correct code for strcpy, that
      is used to copy the contents from src to dest?

      a) strcpy (char *dst,char *src)
            {
            while (*src)
            *dst++ = *src++;
            }
      b) strcpy (char *dst,char *src)
            {
            while(*dst++ = *src++);
            }
      c) strcpy (char *dst,char *src)
            {
            while(*src)
            { *dst = *src;
            dst++; src++;
            }
            }
      d) strcpy(char *dst, char *src)
            {
            while(*++dst = *++src);
            }

            **ans: b ('a'-null character not assigned    'c'-null
            character not assigned  'd'-first character is skipped)**

```
754. main()
        {
        int X,b=7;
        X = b>8 ? b <<3 : b>4 ? b>>1:b;
        printf("%d",X);
        }

        ans: 3

755. main()
        {
        char *src = "Hello World";
        char *dst;
        dst = (char *)malloc(20);
        while(*dst = *src){dst++;src++;}
        printf("%s",dst);
        getch();
        }

        ans: no output

756. main()
        {
        char *src = "Hello World";
        char *dst;
        dst = (char *)malloc(20);
        while(*dst++ = *src++);
        printf("%s",dst);
        getch();
        }

        ans: garbage characters (dst is pointing to the
        character next to the null character)

757. main()
        {
        char *src = "Hello World";
        char *dst;
        while(*dst++ = *src++);
        printf("%s",dst);
        getch();
        }

        ans: error (use of dst before definition. Assign some
    address to dst)

758. main()
        {
        char *src = "Hello World";
        char dst[20];
        while(*dst++ = *src++);
```

```
          printf("%s",dst);
          getch();
          }
```

**ans: error (lvalue required)**

759.  int main()
```
          {
          for(;;);
          printf("Hello\n");
          return 0;
          }
```

**ans: Runs in an infinite loop without printing anything.**

760.  FUNC (int *p)
```
          {
          p = (int *)malloc(100);
          printf("p:%x ",p);
          }
```

      int main()
```
          {
          int *ptr;
          FUNC(ptr);
          printf("Ptr:%x",ptr);
          return 0;
          }
```

**ans: Both print different values (p:882 Ptr:1097)**

761.  int main()
```
          {
          char a[] = "world";
          printf("%d %d\n",strlen(a),sizeof(a));
          return 0;
          }
```

**ans: 5 6**

762.  main()
```
          {
          char *s = "Hello";
          printf("%s",1(s));
          }
```

**ans: error (call of nonfunction)**

763.  main()
```
          {
          char *s = "Hello";
```

```
printf("%s",1[s]);
}
```

**ans: error (it has to print from memory location 9b i.e.   'e')**

764.  main()
```
{
char *s = "Hello";
printf("%s",&1[s]);
}
```

**ans: ello**

765.  **char ( * ( f () ) [] )()**

**ans: f is a function returning pointer to array[] of pointer to function returning char.**

766.  main()
```
{
int i;
i=(2,3);
printf("%d",i);
}
```

**ans: 3**

767.  main()
```
{
char str[]="GESL";
printf("%d %d",sizeof(str),strlen(str));
}
```

**ans: 5 4**

768.  main()
```
{
int i;
for(i=0;i++;i<100)
printf("hello world\n");
}
```

**ans: no output (for loop condition fails)**

769.  main()
```
{
char i;
for(i=1;i++;i<100)
printf("hello world %d\n",i);
}
```

**ans: hello world 1…….hello world 127……hello world –128…...hello world –1…...hello world 0**

770. main()
```
{
int i;
for(i=1;i++;i<100)
printf("hello world %d\n",i);
}
```

**ans: hello world 1…….hello world 32767……hello world –32768....hello world –1…...hello world 0**

771. main()
```
{
char c;
scanf("%s",c);
}
```

**ans: it asks for a character when you type a character it will give error because 99 memory location i.e., 'c' (which is protected memory and not accessible) is used to store typed character.**

772. main()
```
{
int k=5;
for(++k<5 && k++/5 || ++k<8);
printf("%d\n",k);
}
```

**ans: error (for loop syntax error)**

773. main()
```
{
int k=5;
if(++k<5 && k++/5 || ++k<8);
printf("%d\n",k);
}
```

**ans: 7**

774. main()
```
{
int k=5;
if(++k<5 && k++/5 && ++k<8);
printf("%d\n",k);
}
```

**ans: 6**

```
775. main()
        {
        int k=5;
        if(++k<5 || k++/5 && ++k<8);
        printf("%d\n",k);
        }

        ans: 8

776. main()
        {
        int k=5;
        if(++k<5 || k++/5 || ++k<8);
        printf("%d\n",k);
        }

        ans: 7

777. int *func(int a, int b, int *c)
        {
        int x=a+b;
        *c=a-b;
        return(&x);
        }
    main()
        {
        int *ptr1,*ptr2;
        ptr1=(int *)malloc(sizeof(int));
        ptr2=func(20,10,ptr1);
        printf("%d %d\n",*ptr1,*ptr2);
        }

        ans: bug in the code (we are returning address of a auto
        variable whose scope is lost after function returns)

778. int *func(int a, int b, int *c)
        {
        static int x=a+b;
        *c=a-b;
        return(&x);
        }
    main()
        {
        int *ptr1,*ptr2;
        ptr1=(int *)malloc(sizeof(int));
        ptr2=func(20,10,ptr1);
        printf("%d %d\n",*ptr1,*ptr2);
        }
```

**ans: error (illegal initialization of x. since x is a
static variable it should be initialized with constant
expression)**

779.  int *func(int a, int b, int *c)
```
        {
        static int x;
        x=a+b;
        *c=a-b;
        return(&x);
        }
    main()
        {
        int *ptr1,*ptr2;
        ptr1=(int *)malloc(sizeof(int));
        ptr2=func(20,10,ptr1);
        printf("%d %d\n",*ptr1,*ptr2);
        }
```

**ans: 10 30**

780.  int main()
```
        {
        int i=10,j;
        if((j=~i)<i)
        printf ( "True" ) ;
        else
        printf ( "False" ) ;
        }
```

**ans: True**

781.  int main()
```
        {
        int i=10,j;
        if((j=~i)<i)
        printf ( "True" ) ;
        else
        printf ( "False" ) ;
        }
```

**ans: Flase**

782.  int main()
```
        {
        unsigned int i=-10,j=10;
        if(j<i)
        printf ( "True " ) ;
        else
        printf ( "False " ) ;
        printf("%d %u",i,i);
```

```
        }

        ans: True -10 65526

783.  main()
        {
        FILE *fp;
        printf("%d\n",sizeof(fp));
        }

        ans: 4 (pointer takes 4 bytes)

784.  main()
        {
        int a=10,b=20;
        a^=b^=a^=b;
        printf("%d %d\n",a,b);
        }

        ans: 20 10

785.  main()
        {
        int a=10,20;
        int b;
        a^=b^=a^=b;
        printf("%d %d\n",a,b);
        }

        ans: error (declaration error)

786.  main()
        {
        int a,b;
        a=(10,15);
        b=10,15;
        printf("%d %d",a,b);
        }

        ans: 15 10

787.  main()
        {
        int i=10;
        switch(i)
        {
        case 10: printf("Hello ");
        {
        case 1 : printf("World ");
        }
        case 5: printf("Hello World ");
```

```
            }
            }

            ans: Hello World Hello World
```

788.  main()
```
            {
            char str1[]="Hello";
            char str2[]="Hello";
            if ( str1==str2 )
            printf("True\n");
            else
            printf("False\n");
            }

            ans: False
```

789.  main()
```
            {
            # include <stdio.h>
            int i = 10 ;
            printf("%d\n", i/2 );
            }

            ans: 5
```

790.  #pragma pack(2)
```
      struct SIZE
            {
            int i;
            char ch ;
            double db ;
            };
      main()
            {
            printf("%d\n",sizeof(struct SIZE));
            }
```

            **ans: 12 (actually it takes 11 bytes since packing is there it takes 12 bytes)**

791.  main()
```
            {
            int arr[]={ 1,2,3,4 };
            int *ptr ;;;;
            ptr++ = arr;
            printf("%d,%d",ptr[2],arr[2]);
            return 0;
            }

            ans: error (lvalue required)
```

792. main()
```
        {
        char s[10];
        scanf ("%s",s);
        printf(s);
        }
        what is the output if input is abcd
```

**ans: abcd**

793. main()
```
        {
        char c = 255;
        printf ("%d",c);
        return 0;
        }
```

**ans: -1**

794. main()
```
        {
        int i;
        for (i=7;i<=0;i--)
        printf ("hello\n");
        }
```

**ans: no output (for loop codition fails on first
iteration)**

795. main()
```
        {
        printf( printf ("world") );
        }
```

**ans: error (printf(5) gives error. Since memory location
5 is not accessible)**

796. main()
```
        {
        scanf("%d");
        printf();
        }
```

**ans: error (too few parameters in call to printf)**

797. main()
```
        {
        scanf("%d");
        printf("manu");
        }
```

**ans: manu (whatever you type for scanf output will be manu)**

798. #define islower(c) ('a'<=(c) && (c)<='z')
     #define toupper(c) (islower(c)?(c)-('a'-'A'):(c))
     main()
          {
          char *p="i am fine";
          while(*p)
          printf("%c",toupper(*p++));
          }

     **ans: AFE (macro substitution 3 times)**

799. main()
          {
          200;
          printf("tricky problem");
          }

     **ans: tricky problem**

800. which is the null statement?
     a) ;
     b) {}
     c) '\0';
     d)all of these

     **ans: a)**

801. what is the correct prototype of printf function ?
     a)printf(char *p,...);
     b)printf(const *char *p,...);
     c)printf(const char *p,...);
     d)printf(const *char p,...);

     **ans: c)**

802. For a linked list implementation which searching technique is not
     applicable?
     a)linear search
     b)none
     c)quick sort
     d)binary search

     **ans: d)**

803. what is big-endian.
     a) MSB at lower address LSB at higher address

b) LSB at lower address MSB at higher address
c) memory mgmt technique
d) none of the above

**ans: a)**

804. what is Little-endian.
    a) MSB at lower address LSB at higher address
    b) LSB at lower address MSB at higher address
    c) memory mgmt technique
    d) none of the above

**ans: b)**

805. what is the scheduling algorithm used in general operating
    systems.
    a) FCFS algorithm
    b) Highest Priority First algorithm
    c) Round-Robin algorithm
    d) None of the above

**ans: c)**

806. void main()
```
{
char *mess[]={"Have","a","nice","day","Bye"};
printf("%d  %d",sizeof(mess),sizeof(mess[1]));
}
```

**ans: 20 4 (mess is an array of 5 pointers and mess[1] is
pointer. Here pointer takes 4 bytes)**

807. void main()
```
{
int i,count=0;
char *p1="abcdefghij";
char *p2="alcmenfoip";
for(i=0;i<=strlen(p1);i++)
{
if(*p1++ == *p2++)
count+=5;
else
count-=3;
}
printf("count=%d\n",count);
}
```

**ans: count=6**

808. what does main return on successful execution?
    a. 1

```
   b. 0
   c. -1
   d.Nonzero
```

**ans: b**

809. 
```
main(int argc,char *argv[])
     {
     printf((argc > 1 ? "%c" : "%c",*++argv);
     }
     If the i/p string is "GESL Bangalore".
```

**ans: B (check it out)**

810. How do u declare a pointer to an array of pointers to int?
```
a. int *a[5];
b. int **a[5];
c. int *(*a)[5];
d. u con not declare
```

**ans: c**

811. 
```
main()
     {
     int a;
     char *p;
     a = sizeof(int) * p;
     printf("%d\n",a);
     }
```

**ans: illegal use of pointer (pointer multiplication is invalid)**

812. 
```
#define SIZE sizeof(int)
main()
     {
     int i=-1;
     if( i < SIZE )
     printf("True\n");
     else
     printf("False\n");
     }
```

**ans: True**

813. **int (*fun())[]**

**ans: function returning a pointer to an array of integers**

814. main()

```
{
int a=8,d;
int *p;
p=&a;
d=a/*p;
printf("%d\n",d);
}
```

**ans: error (there should be space between / and *)**

815. main()
```
{
int a=8,d;
int *p;
p=&a;
d=a/ *p;
printf("%d\n",d);
}
```

**ans: 1**

816. main()
```
{
char *a="Hello";
a++ = 'h';
printf("%s\n",a);
}
```

**ans: error (lvalue required. Both assignment and increment is on a)**

817. main()
```
{
char *a="Hello";
*a++ = 'h';
printf("%s\n",a);
}
```

**ans: ello (here assignment is to *a and increment is on a)**

818. main()
```
{
char p[]="Hello";
p[0]='h';
printf("%s\n", p);
}
```

**ans: hello**

819. #define mysizeof(a) (&a+1) – &a

```
main()
    {
    float d;
    printf("%d ", &d);
    printf("%d ", &d+1);
    printf("%d ",mysizeof(d));
    printf("%d",&d+1-&d);
    }

    ans: 9216 9220 1 1
```

820.  ```
main()
    {
    int *p=10;
    printf("%d\n",*p);
    }
```

ans: error (value at memory location 10 which is not accessible)

821.  ```
main()
    {
    int *p=10;
    printf("%d\n",p);
    }
```

ans: 10

822.  ```
main()
    {
    int i=-1;
    i<<=2;
    printf("%d\n",i);
    }
```

ans: -4

823.  ```
main()
    {
    int i= 0xffffffff;
    printf("%d\n",i);
    }
```

ans: -1

824.  ```
main()
    {
    int A=1,B=2;
    if(A==B < printf("Hello "))
    printf("world\n");
    else
```

```
            printf("Bangalore\n");
            }

            ans: Hello world (< has highest priority than ==)
```

825. main()
```
            {
            int i;
            for(i=0; i< 10; i++)
            {
            int j=10;
            j++;
            printf("j= %d\n", j);
            }
            }
```

**ans: j= 11 will be printed 10 times**

826. union test
```
            {
            int a;
            union test *p;
            };
        main()
            {
            union test q;
            printf(" a= %d\n ", q.a);
            }
```

**ans: a= garbage value**

827. register int a,b;
```
        main()
            {
            for(a=0 ; a<5 ; a++)
            b++;
            }
```

**ans: error (storage class 'register' is not allowed
here)**

828. #define dprint(expr) printf(" expr= %d \n ", expr)
```
        main()
            {
            int i=10,j=2;
            dprint(i/j);
            }
```

**ans: expr= 5**

829. main()

```
{
int *p ;
p=(int *)malloc(-10);
printf("%d",p);
free(p);
}
```

**ans: 0 (no space is allocated for p. p is a null pointer)**

830. main()
```
{
int *p ;
p=(int *)malloc(10);
printf("%d",p);
free(p);
}
```

**ans: 2266 (starting address of the allocated block)**

831. main()
```
{
for(printf("a");printf("b");printf("c"));
}
```

**ans: abcbcbcbcbcb……. Infinite loop**

832. fun()
```
{
return 10 ;
}
```
main()
```
{
int i= 10 * fun() ;
printf("%d",i);
}
```

**ans: 100**

833. fun()
```
{
return 10 ;
}
```
**int i= 10 * fun() ;**
main()
```
{
printf("%d",i) ;
}
```

ans: illegal initialization error (static and global
variables should be initialized with constant or
constant expression)

834.  main()
```
{
int i=100 ;
printf("%d ", sizeof(i++));
printf("%d ",i) ;
}
```

ans: 2 100 (sizeof operator operand will not be
evaluated)

835.  main()
```
{
int i=100 ;
printf("%d ", sizeof(++i);
printf("%d ",i) ;
}
```

ans: 2 100 (sizeof operator operand will not be
evaluated)

836.  main()
```
{
int i=100 ;
printf("%d ", sizeof(++i++));
printf("%d ",i) ;
}
```

ans: error (lvalue required and not allowed type for
sizeof operator)

837.  Which one of the following data structures is best suited for
searching ?

a) Arrays
b) Singly Linked List
c) Doubly Linked List
d) Hash Table

ans: d)

838.  Which of the following data structures is best suited for
Deletion ?

a) Arrays
b) Singly Linked List
c) Doubly Linked List
d) Hash Table

839. Which one of these is not a scheduling technique in Operating System?

    a) Last-Come-First-Serve Scheduling
    b) First-Come-First-Serve Scheduling
    c) Preemptive Scheduling
    d) Round Robin Scheduling

        **ans: a)**

840. "Banker's Algorithm" is used for :

    a) Deadlock Detection
    b) Deadlock Avoidance
    c) Deadlock Prevention
    d) All of the above

        **ans: b)**

841. main()
```
        {
        int a = 1;
        #define p a
        printf("%d %d ",a++,p++);
        }
```

        **ans: 2 1**

842. main()
```
        {
        #include<stdio.h>
        int a = 90 ;
        printf("%d",a) ;
        }
```

        **ans: 90**

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

        **ans: executes until the stack overflows**

844. #define max "hello"
    main()
```
        {
        printf(max);
```

```
        }

        ans: hello

845.  #define max main()
      main()
        {
        max;
        printf("hello wolrd\n ");
        }

        ans: executes until the stack overflows

846.  typedef int *p ;
      main()
        {
        int a = 90 ;
        p p1 ;
        p1 = &a ;
        printf("%d",*p1) ;
        }

        ans: 90

847.  main()
        {
        int i=1 ;
        printf(i ?"one" : "zero") ;
        }

        ans: one

848.  main()
        {
        int i=1;
        printf("%d",i ? 1 : 0) ;
        }

        ans: 1

849.  main()
        {
        int a=90,b=100;
        a++;
        a=(a ^ b) ^ (a = b );
        b = a^b^a ;
        --a ;
        printf("%d %d",a++,b++) ;
        }

        ans: 90 100
```

```
850.  main()
          {
          int a = 10 , b = 100 ;
          swap(&a , &b) ;
          printf("%d %d",a,b) ;
          }
      swap(int *a , int *b)
          {
          *a = *a + *b ;
          *b = *a - *b ;
          *a = *a - *b ;
          swap1(&a , &b) ;
          }
      swap1(int **a , int **b)
          {
          **a = **a + **b ;
          **b = **a - **b ;
          **a = **a - **b ;
          }

          ans: 10 100

851.  main()
          {
          void *ptr ;
          int a = 10 ;
          ptr = &a ;
          printf("%d",*ptr) ;
          }

          ans: error (indirection operator * should not be applied
          on void pointer. Since compiler does not know the size
          of the operand which void pointer is pointing to)

852.  main()
          {
          void *ptr ;
          int a = 90 ;
          char *ptr1 = "hello" ;
          ptr = a ;
          ptr = ptr1 ;
          }

          ans: executes without any error

853.  main()
          {
          char *p = "helloo" ;
          char *p1 = "strcat" ;
          while((*(p++) = *(p1++)) != '\0')
```

```
        {
        ;
        }
        }
```

**ans: contents are copied**

854. ```
     int g = 10 ;
     main()
        {
        int g = 10 ;
        printf("%d",g) ;
        }
        int g ;
```

**ans: 10**

855. ```
     int g = 10 ;
     main()
        {
        extern int g;
        printf("%d",g) ;
        }
        int g ;
```

**ans: 10**

856. ```
     //int g = 10 ;
     main()
        {
        extern int g;
        printf("%d",g) ;
        }
        int g ;
```

**ans: 0**

857. ```
     main()
        {
        int a = 1 ;
        int b = 0 ;
        a = a++ + --b * a++ ;
        printf("%d",a) ;
        }
```

**ans: 2**

858. ```
     struct s
        {
        int si;
        union u
```

```
            {
            float uf;
            char uc;
            };
            };
      main()
            {
            printf("%d",sizeof(struct s));
            }

            ans: declaration terminated incorrectly


859.  struct s
            {
            int si;
            union u
            {
            float uf;
            char uc;
            }a;
            };
      main()
            {
            printf("%d",sizeof(struct s));
            }

            ans: 6


860.  struct st
            {
            int a;
            char b;
            }
      main()
            {
            }

            ans: struct st is return type of main (since statement
            termination is not there for struct template)


861.  typedef struct info
            {
            int i;
            char b;
            }node;
      main()
            {
            struct info node1;
            node1.i=55;
            printf("%d",node1.i);
            }
```

**ans: 55 (node is different from node1)**

862.  struct a
```
       {
       int i;
       int display()
       {
       printf("hello world\n");
       }
       };
```
      main()
```
       {
       strcut a vara;
       vara.display();
       }
```

        **ans: functions may not be a part of a struct or union**

863.  struct a
```
       {
       int (*ptr)();
       };
       int display()
       {
       printf("Global Edge\n");
       }
```
      main()
```
       {
       struct a structa;
       structa.ptr=display;
       structa.ptr();
       }
```

        **ans: Global Edge (through function pointers functions
        can be implemented in structures)**

864.  typedef int *ABC;
      typedef ABC XYZ[10];
      int main()
```
       {
       XYZ var;
       }
```
       1. var is an array of integer pointers.
       2. var is a pointer to an integer array.

        **ans: only 2 is correct**

865.  union tag
```
       {
       int a;
```

```
        char x;
        char y;
        }name;
int main()
        {
        name.a=258;
        printf("\n x = %d y = %d ",name.x,name.y);
        }
```

**ans: x = 2 y = 2**

866. int main()
```
        {
        int a[20];
        int *p,*q,val;
        p = &a[0];
        q = &a[10];
        val = q - p;
        printf("p %d ",p);
        printf("q %d ",q);
        printf("val %d",val);
        }
```

**ans: p 8640 q 8660 val 10**

867. struct key
```
        {
        char *word[2];
        int count;
        char c;
        }abc;

int main()
        {
        printf("\nsize %d",sizeof(abc));
        }
```

**ans: size 11 (pointer takes 4 bytes)**

868. main()
```
        {
        int a;
        fun();
        printf("%d",a);
        a=50;
        }
fun()
        {
        int i;
        *(&i+4) = 100;
        }
```

```
           ans: error (&i+4 memory location is not allocated and we
           are trying to assign a value to this memory location)
```

869.  main()
```
           {
           #define x 5
           int b;
           b = x;
           printf("%d",b);
           }

           ans: 5
```

870.  main()
```
           {
           int a; #define y 10
           a=y;
           printf("%d",a);
           }
```

           ans: #define (should come at the beginning of  the
     block)

871.  #define s -50
     main()
```
           {
           int s;
           #ifdef s
           printf("Hell\n");
           #else
           printf("Heaven\n");
           #endif
           }
```

           ans: error (declaration terminated incorrectly i.e  int
     -50;)

872.  #define s -50
     main()
```
           {
           int a;
           #ifdef s
           printf("Hell\n");
           #else
           printf("Heaven\n");
           #endif
           }

           ans: Hell
```

873.  How many times can a comment be nested ?

    A)COMMENT_NEST_LIMIT times
    B)COMMENT_LIMIT times
    C)ONE time
    D)Not even Once

        **ans: D)**

874.  main()
        {
        int i,j;
        i = 06;
        **j = 09;**
        printf ("%d %d\n",i,j);
        }

        **ans: error (illegal octal digit. 9 is not there in octal system)**

875.  main()
        {
        int i,j;
        **i = o6;**
        **j = 09;**
        printf ("%d %d\n",i,j);
        }

        **ans: error (illegal octal digit. 9 is not there in octal system. Octal number starts with 0,zero not with letter o)**

876.  **# undef __FILE__**
    # define __FILE__  "GLOBALEDGE"
    main()
        {
        printf("%s\n",__FILE__);
        }

        **ans: Bad undef directive syntax**

877.  # define LINE
    # define NAME "GESL"
    main()
        {
        printf("%d %s\n",LINE,NAME);
        }

        **ans: error (LINE is not defined)**

878.  # define LINE 1
    # define NAME "GESL"

```
main()
    {
    printf("%d %s\n",LINE,NAME);
    }
```

**ans: 1 GESL**

879. ```
main()
    {
    int i=10;
    float j=2.5;
    printf("%d ",sizeof(j+++i++));
    printf("%d %f",i,j);
    }
```

**ans: 4 10 2.500000**

880. ```
int main()
    {
    int i = 5;
    if(1)
    {
    static int i;
    i++;
    printf("%d ", i);
    }
    printf("%d", i);
    }
```

**ans: 1 5**

881. ```
int main()
    {
    int a[4] = {23, 67, 90};
    printf("%d", a[3]);
    }
```

**ans: 0 (when there are fewer initializations remaining
elements are zero)**

882. ```
int main()
    {
    int i = 1, 2;
    printf("%d", i);
    }
```

**ans: error (declaration terminated incorrectly)**

883. ```
int main()
    {
    int i;
```

```
        for( i=0;;i++)
        {
        i = i+2;
        break;
        printf("%d", i);
        }
        }
```

**ans: no output (for loop enters only once and after
i=i+2 it breaks )**

884.  int main()
```
        {
        int i;
        i = 1, 2;
        printf("%d", i);
        }
```

**ans: 1**

885.  int i =20;
```
      int maxlen = i;
      int main()
        {
        int j = i;
        printf("i=%d , j=%d\n", i , j);
        }
```

**ans: illegal initialization error (static and global
variables shoul be initialized with constants or
constant expression)**

886.  int main()
```
        {
        int i =10;
        printf("%d", k);
        printf("%d",i);
        }
        int k = 20;
```

**ans: error (undefined symbol k)**

887.  int main()
```
        {
        int i =10;
        extern int k;
        printf("%d ", k);
        printf("%d",i);
        }
        int k = 20;
```

**ans: 20 10**

888. 
```c
int i =20;
int i,j=10;
int i;
main()
    {
    int j =20;
    printf("i=%d , j=%d\n", i, j);
    }
```

**ans: i=20 , j=20**

889. 
```c
int main()
    {
    int k=2,i =10;
    while(k--)
    {
    printf("%d ",disp(i));
    }
    }
    disp(int k)
    {
    static int i=0;
    return i=i+k;
    }
```

**ans: 10 20**

890. header files **usually** contains
    a)only definitions
    b)only declarations
    c)both
    d)compiled code for functions

**ans: b)**

891. 
```c
int main()
    {
    int i =3;
    while(i--)
    {
    int i =10;
    printf("%d ",i);
    }
    }
```

**ans: 10 10 10**

892. 
```c
int main()
    {
```

```
            char s[] = "hello\0 world";
            printf("%s...%d",s,strlen(s));
            }
```

**ans: hello…5**

893.   int main()
```
            {
            printf("%%%    s","hello");
            }
```

**ans: %hello**

894.   What does fgetc return

       (a) char
       (b) int
       (c) unsigned int
       (d) void

**ans: (b)**

895.   main()
```
            {
            int i = 24;
            printf("%xd",i);
            }
```

**ans: 18d**

896.   main()
```
            {
            int i = 24;
            printf("%0xd",i);
            }
```

**ans: 18d**

897.   struct node
```
            {
            int i;
            };
       main()
            {
            struct node n1;
            printf("%d",n1.i);
            }
```

**ans: garbage value**

898.   struct node_tag

```
        {
        int i;
        struct node_tag *pt;
        };
    main()
        {
        printf("%d",sizeof(node_tag));
        }
```

**ans: error (struct keyword is missing)**

899.  struct node_tag
```
        {
        int i;
        struct node_tag *pt;
        };
    main()
        {
        printf("%d",sizeof(struct node_tag));
        }
```

**ans: 6**

900.  typedef struct node_tag
```
        {
        int i=0;
        int j;
        }node;

    main()
        {
        node n1;
        printf("%d",n1.i);
        }
```

**ans: error (i should not be initialized like that)**

901.  struct
```
        {
        int i;
        }node ;
    main()
        {
        printf("%d",node.i);
        }
```

**ans: 0**

902.  main()
```
        {
        struct
```

```
        {
        int i;
        }node ;
        printf("%d",node.i);
        }
```

**ans: 19125 (garbage value)**

903.  struct tag
```
        {
        int i;
        };
main()
        {
        struct tag node;
        printf("%d",node.i);
        }
```

**ans: garbage value (19125)**

904.  struct node_tag
```
        {
        int a;
        struct node_tag *pt;
        };

main()
        {
        struct node_tag n1;
        n1.pt=&n1;
        n1.pt->a=5;
        printf("%d",n1.a);
        }
```

**ans: 5**

905.  main()
```
        {
        int n;
        scanf("%d",n);
        }
```

**ans: runtime error (if n value equals address of inaccessible memory location)**

906.  (void *) is called

(a)pointer to void
(b)pointer to any data type
(c)generic pointer
(d)None of the above

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

**ans: 27**

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

**ans: 30**

909.  int main()
```
{
char *p = "Welcome To GESL\n";
*(p+10);
fprintf(stderr,"%s",p);
return 'c';
}
```

**ans: Welcome To GESL**

910.  int main()
```
{
char *p = "Welcome To GESL\n";
*(p+++10);
fprintf(stderr,"%s",p);
return 'c';
}
```

**ans: elcome To GESL**

911.  int main(void)
```
{
puts("hello\0world");
}
```

**ans: hello (\0 null character is there after hello)**

912.  union u
```
{
int ival;
float fval;
```

```
       char *sval;
       }
       size of u is?

       ans: 4 bytes
```

913.  struct x
```
       {
       int i; int j;int k;
       };

       struct x *p;
       struct x arr[3];
       p =&arr[0];
       p++;
       what is p pointing to?
       a) pointing to i of arr[0]
       b) pointing to j of arr[0]
       c) pointing to k of arr[1]
       d) pointing to i of arr[1]
```

       **ans: d)**

914.  struct a
```
       {
       int b;
       };
   struct b
       {
       int b;
       };
   int main()
       {
       struct a first;
       struct b second;
       first.b =10;
       second = first;
       printf("%d",second.b);
       }
```

       **ans: error (second and first are two different structure variables)**

915.  struct a
```
       {
       int b;
       };

   int main()
       {
```

```
        struct a first,second;
        first.b =10;
        second = first;
        printf("%d",second.b);
        }

        ans: 10 (second and first variables belong to same
structure)

916.  struct a
        {
        int x;
        float y;
        double z;
        struct a b;
        };

int main()
        {
        ;
        }

        ans: error (undefined structure 'a')

917.  struct a
        {
        int x;
        float y;
        double z;
        struct a *b;
        };

int main()
        {
        ;
        }

        ans: no error

918.  struct a
        {
        struct b
        {
        int a;int b;
        }c;
        int *ptr;
        }d;

int main()
        {
        d.ptr=&d.c.a;
```

```
        }

        ans: no error

919.  int main(void)
        {
        int *intPtr ;
        intPtr = (char*)malloc(sizeof(10));
        printf("\n The starting address is %d \n ",intPtr);
        return 0;
        }

        ans: The starting address is 2274

920.  int main(void)
        {
        int intNum1,intNum2,num = 1,i;
        printf("\nEnter first number \n");
        scanf("%d",&intNum1);
        printf("\nEnter second number \n");
        scanf("%d",intNum2);
        for(i = 0;i<=3;i++)
        {
        num = intNum1 * intNum2 * num;
        }
        printf("\n num = %d " , num);
        return 0;
        }

        ans: error (second scanf function reads a value into a
        memory location which may not be user accessible some
        times)

921.  int main(void)
        {
        int a=1,b=0, x;
        x = a++ && ++b;
        printf("%d %d %d ",a,b,x );
        }

        ans: 2 1 1

922.  char *fn();
      main()
        {
        char *s;
        s = fn();
        printf("%s\n",s );
        }
        char *fn()
        { return "Hello"; }
```

923. main()
```
{
int i;
for( i=0; i<10-1; i+=2 );
i+= 2;
printf("i = %d\n", i );
}
```

**ans: i = 12**

924. f()
```
{ return 1,2,3; }
```

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

**ans: 3**

925. What is the difference between ++*ip and *ip++ ?

a) both increment value
b) ++*ip increment value and *ip++ increment address
c) both increment address
d) ++*ip increment address and *ip++ increment value

**ans: b)**

926. int main (void)
```
{
int x = 48;
printf("x = %s\n", x );
}
```

**ans: error (memory location 48 is not user accessible)**

927. # define ONE 1
```
# define TWO 2
//# define ONE TWO
//# define TWO ONE
int main (void)
{
printf("ONE = %d, TWO = %d\n", ONE, TWO );
}
```

**ans: ONE = 1, TWO = 2**

928. 
```
# define ONE 1
# define TWO 2
# define ONE TWO
//# define TWO ONE
int main (void)
      {
      printf("ONE = %d, TWO = %d\n", ONE, TWO );
      }
```

**ans: ONE = 2, TWO = 2**

929. 
```
# define ONE 1
# define TWO 2
# define ONE TWO
# define TWO ONE
int main (void)
      {
      printf("ONE = %d, TWO = %d\n", ONE, TWO );
      }
```

**ans: error (undefined symbol ONE and TWO)**

930. If the command line arguments for the following program are <a.out>
   and <GlobalEdgeSoftwareLtd>, what is the output of the program ?

```
int main(int argc, char **argvar)
      {
      printf("output = %s\n", *argvar[1]);
      }
```

**ans: runtime error (check it out)**

931. 
```
void fun( int, int );
int main ( void )
      {
      fun( 12, ( 13, ( 14, 17 ) ) );
      return 0;
      }

void fun( int x, int y )
      {
      printf("x = %d, y = %d\n", x, y );
      }
```

**ans: x = 12, y = 17**

932. main()

```
        {
        int i,j;
        int arr[4][4] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
        for (i=2;i<0;i--)
        for (j=2;j<=0;j--)
        printf("%d", arr[i][j]);
        }
```

**ans: no output**

933.  void main()
```
        {
        int i,x,sum=0;
        int arr[6]={1,2,3,4,5,6};
        for (i=0;i<4;i++)
        sum += func(arr[i]);
        printf("%d", sum);
        }
func(int x)
        {
        int val,x;
        val = 2;
        return(x+ val++);
        }
```

**ans: error (multiple declaration of x)**

934.  Where is a variable defined in a function stores?

**ans. Process Swappable Area**

935.  void main()
```
        {
        int ari[] = {1,2,3,4,5};
        char arc[] = {'a','b','c','d','e'};
        printf("%d ",&ari[4]-&ari[2]);
        printf("%d ",&arc[3]-&arc[0]);
        }
```

**ans: 2 3**

936.  void main()
```
        {
        int i=0,j=0;
        int arr[4][4] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
        clrscr();
        for (i=2;i>=0;i--)
        for(j=2;j>=0;j--)
        printf("%d ", *(*(arr+j)+i));
```

```
            getch();
            }

            ans: 11 7 3 10 6 2 9 5 1

937.  void main()
            {
            int a=10,b=11;
            printf("%d ",a+++b);
            printf("%d",a+++b);
            }

            ans: 21 22

938.  void main()
            {
            int a;
            void c;
            }

            ans: error (size of c is unknown)

939.  void main()
            {
            int a;
            void *c;
            }

            ans: no error

940.  void main()
            {
            int a,b;
            a=0;
            b=(a=0)?2:3;
            printf("%d",b);
            }

            ans: 3

942.  f1(int c)
            {
            printf("%d", c);
            }
      main()
            {
            int a=2;
            f1(a++);
            }

            ans: 2
```

```
943.  f(int t)
          {
          switch(t)
          {
          int c;
          case 2: c=3;
          case 3: c=4;
          case 4: c=5;
          case 5: c=6;
          default: c=0;
          }
          printf("%d",c);
          }
      main()
          {
          f(3);
          }

          ans: error (undefined symbol 'c')

944.  f(int t)
          {
          int c;
          switch(t);
          {
          case 2: c=3;
          case 3: c=4;
          case 4: c=5;
          case 5: c=6;
          default: c=0;
          }
          printf("%d",c);
          }
      main()
          {
          f(3);
          }

          ans: error (case outside of switch since switch is
      terminated by ; )

945.  f(int t)
          {
          int c;
          switch(t)
          {
          case 2: c=3;
          case 3: c=4;
          case 4: c=5;
          case 5: c=6;
```

```
        default: c=0;
        }
        printf("%d",c);
        }
main()
        {
        f(3);
        }
```

**ans: 0**

946. What is the fallacy in the following program segment?

```
int *f1()
        {
        int a=5;
        return &a;
        }
f()
        int *b=f1()
        int c=*b;
        }
```

**ans: we should not return address of a auto variable as its scope will be lost when function returns**

947. Give the C language equivalents of the following
     a)Function returning an int pointer
     b)Function pointer returning an int pointer
     c)Function pointer returning an array of integers
     d)Array of function pointer returning an array of integers

**int \*x();**
**int \*(\*x)();**
**int ( (\*x)() )[];**
**int ( (\*x[])() )[];**

948. Bootstrap loader program is a program belonging to

     (a) ROM startup software
     (b) ROM extension software
     (c) ROM BIOS software
     (d) ROM Basic software

**ans: (a)**

949. void main()
        {
        int a=3,b=4,c=5;
        a=b+c;
        c=a+b;

```
        b=a+c;
        printf("%d %d %d ",a+b,b+c,c+a);
        a=b*c;
        c=a*b;
        printf("%d %d",a,c);
        }

        ans: 31 35 22 286  6292

950.  void main()
        {
        printf("\nab\bcd\ref");
        }

        ans: efd (\n-new line  \b-backspace \r-carriage return)

951.  struct a
        {
        char b[7];
        char *s;
        };
    struct b
        {
        char *t;
        struct a y;
        };
    main()
        {
        struct b q={"Raipur" , "Kanpur" , "Jaipur"};
        printf("%s %s " , q.t , q.y.s);
        printf("%s %s" ,++q.t , ++q.y.s);
        }

        ans: Raipur Jaipur aipur aipur

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

        ans: 1,2

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

        ans: 1 2 3 4 5 6 7 8 9 10 11
```

```
954. main()
        {
        int a[]={10,20,30,40,50};
        fun(a+1);
        }
        fun(int *p)
        {
        for(int i=1;i<=3;i++)
        printf("%d",*(p+i));
        }

        ans: error (i should be declarated before for loop)

955. main()
        {
        int a[]={10,20,30,40,50};
        fun(a+1);
        }
        fun(int *p)
        {
        int i;
        for( i=1;i<=3;i++)
        printf("%d",*(p+i));
        }

        ans: 30 40 50

956. main()
        {
        enum day {saturday,
        sunday=3,
        monday,
        tuesday
        };
        printf("%d %d",saturday,tuesday);
        }

        ans: 0 5

957. main()
        {
        int x;
        enum day {
        saturday,
        sunday=-1,
        monday,
        tuesday
        };
        x=monday;
        printf("%d",x);
        }
```

958.  #define ADD(X,Y) X+Y
      main()
            {
            #undef ADD(X,Y)
            fun();
            }
      fun()
            {
            int y=ADD(3,2);
            printf("%d",y);
            }

            ans: error (linker error)

959.  #define ADD(X,Y) X+Y
      main()
            {
            //#undef ADD(X,Y)
            fun();
            }
      fun()
            {
            int y=ADD(3,2);
            printf("%d",y);
            }

            ans: 5

960.  int x;
      int *p;
      int **p1;
      int ***p2;
      How to assign each one?

            ans:  p=&x;
                  p1=&p;
                  p2=&p1;

961.  Which of the following is illegal
         (a)void v;
         (b)void *v;
         (c)void **v;
         (d)all are legal

            ans: (a)

962.  #define int INTEGER/*line1*/
      #define INTEGER int/*line 2*/

```
main()
    {
    INTEGER p=10;/*line 5*/
    printf("%d",p);
    }

    ans: error (undefined symbol INTEGER and undefined
symbol p)

963.  main()
    {
    char str={'H','E','L','L','O','\0'};
    printf("%s/n",str+1);
    }

    ans: error

964.  main()
    {
    char arr[5]={'a','a','b','c','d','e'};
    printf("%s",arr);
    }

    ans: error (too many initializers)

965.  main()
    {
    printf("\% ");
    printf("\\% ");
    printf("%% ");
    printf("%%%%");
    }

    ans: %  \%    %    %%

966.  main()
    {
    printf("%%%% ");
    printf("%%%%% ");
    printf("%");
    }

    ans: %%%  %%%  %

967.  main()
    {
    int i=3;
    while(i>=0)
    printf("%d ",i--);
    return(0);
    }
```

**ans: 3 2 1 0 (loop is executed 4 times)**

968.  main()
```
{
int i=10;
printf("%d %d %d ",i,++i,i++);
}
```

        **ans: 12 12 10**

969.  main()
```
{
int x,y,z;
x=2;
y=5;
z=x+++y;
printf("%d %d %d",x,y,z);
}
```

        **ans: 3 5 7**

970.  void xyz(char a[10])
```
{
int i;
char b[10];
i=sizeof(a);
printf("%d",i);
}
```

    main()
```
{
char s[10];
xyz(s);
}
```

        **ans: 4 (pointer takes 4 bytes)**

971.  void xyz(char a[10])
```
{
int i;
char b[10];
i=sizeof(b);
printf("%d",i);
}
```

    main()
```
{
char s[10];
xyz(s);
}
```

```
        ans: 10

972.  main()
        {
        int i=6;
        printf("%d",i++*i++);
        }

        ans: 42

973.  main()
        {
        char str[20] = "SANJAY";
        printf("%d %d",sizeof(str),strlen(str));
        }

        ans: 20 6

974.  main()
        {
        unsigned int i=3;
        while( i >=0)
        printf( "%d", i--);
        }

        ans: infinite loop

975.  # define swap(a,b) temp=a; a=b; b=temp;
      main()
        {
        int i, j, temp;
        i=5;
        j=10;
        temp=0;
        if( i > j)
        swap( i, j );
        printf( "%d %d %d", i, j, temp);
        }

        ans: 10 0 0

976.  func()
        {
        static int i = 10;
        printf("%d",i);
        i++;
        }

        What is the value of i if the function is called twice?
```

```
            ans: 12

977.
            func(int *i, int*j)
            {
            *i=*i * *i;
            *j=*j* *j;
            }

       main()
            {
            int i = 5, j = 2;
            func(&i,&j);
       printf("%d %d", i, j);
            }

            ans: 25 4

978.  void f(char *p)
            {
            p=(char *) malloc(6);
            strcpy(p,"hello");
            }

       void main()
            {
            char *p="bye";
            f(p);
            printf("%s",p);
            }

            ans: bye

979.  int x(char *a)
            {
            a=(char *) malloc(10*sizeof(char));
            *a="hello";
            }

       main()
            {
            char *a="new";
            x(a);
            printf("%s",a);
            }

            ans: error (nonportable pointer conversion)

980.  main()
            {
            int i =1;
```

```
            switch(i)
            {
            printf ("first");
            i++;
            case 1 : printf ("second");
            break;
            case 2 : printf("");
            break;
            default : printf("");
            break;
            }
            }

            ans: second (first won't be printed)

981.  void main()
            {
            char *s[10]={"welcome","to","india"};
            printf("%d",sizeof(s));
            }

            ans: 40

982.  void main()
            {
            const int i=10;
            int *p;
            p=&i;
            (*p)++;
            printf("\n %d",i);
            return;
            }

            ans: 11 (constant can be modified through a poiter)

983.  void main()
            {
            char c[]="123456789";
            int i=4;
            printf("%c %c", c[i], i[c]);
            }

            ans: 5 5

984.  void main()
            {
            int *ptr;
            p=0;
            p++;
            printf("%u", p);
            }
```

**ans: error (assigning an absolute address to a pointer variable is invalid)**

985. void main()
```
{
double i=0.0;
switch(i)
{
case 0.0:
printf("jgdj");
case 1.0:
printf("ptoy");
break;
default:
printf("hdfv");
}
}
```

**ans: error (switch expression should be integer expression or characters and case values should be constants or constat expression)**

986. void main()
```
{
int a=2;
if(a=3!=3)
printf("3");
else
printf("2");
return;
}
```

**ans: 2**

987. #define TRUE 0
main()
```
{
int i=0;
while(TRUE)
{
printf(" %d \n",i);
i++;
}
printf(" %d \n",i);
i++;
}
```

**ans: 0**

988. main()

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

**ans: 4**

989. f(char *p)
```
{
p[0]? f(++p):1;
printf("%d ",*p);
}
```
main()
```
{
f("abcde");
}
```

**ans: 0 0 101 100 99 98**

990. f(char *p)
```
{
p[0]? f(++p):1;
printf("%c ",*p);
}
```
main()
```
{
f("abcde");
}
```

**ans: null null e d c b (first two are null characters)**

991. f(char *p)
```
{
p=(char *)malloc(sizeof(6));
strcpy(p,"HELLO");
}
```
main()
```
{
char *p="BYE";
f(p);
printf("%s",p);
}
```

**ans: BYE**

992. **f(char **p)**
```
{
*p=(char *)malloc(sizeof(6));
```

```
        strcpy(*p,"HELLO");
        }
main()
        {
        char *p="BYE";
        f(p);
        printf("%s",p);
        }

        ans: HELLO
```

993.  main()
```
        {
        char str[5]="hello";
        if(str==NULL) printf("string null");
        else printf("string not null");
        }
```
**ans: string not null**

994.  void f(int x)
```
        {
        int i;
        for (i=0;i<16;i++)
        {
        if(x &0x8000>>i) printf("1");
        else printf("0");
        }
        }
```
**ans: binary representation of x**

995.  void f(int *p)
```
        {
        static val=100;
        val=&p;
        }
main()
        {
        int a=10;
        printf("%d ",a);
        f(&a);
        printf("%d ",a);
        }
```
**ans: error (nonportable pointer conversion)**

996.  struct a
```
        {
        int x;
        float y;
```

```
        char c[10];
        };
union b
    {
    int x;
    float y;
    char c[10];
    };

main()
    {
    printf("%d %d",sizeof(a),sizeof(b));
    }
```

**ans: error (here sizeof operator operand should be type name not tag name)**

997. 
```
struct a
    {
    int x;
    float y;
    char c[10];
    };
union b
    {
    int x;
    float y;
    char c[10];
    };

main()
    {
    printf("%d %d",sizeof(struct a),sizeof(union b));
    }
```

**ans: 16 10**

998. 
```
main()
    {
    char a[10]="hello";
    strcpy(a,'\0');
    printf("%s",a);
    }
```

**ans: error (0 memory location can't be copied to array a)**

999. 
```
main()
    {
    char a[10]="hello";
    strcpy(a,"\0");
```

```c
        printf("%s",a);
        }
```

**ans: no output**

1000. void f(int*j)
```c
        {
        int k=10;
        j= &k;
        }
    main()
        {
        int i,*j;
        i=5;
        j=&i;
        printf("i=%d ",i);
        f(j);
        printf("i=%d",i);
        }
```

**ans: i=5 =5**

1001. main()
```c
        {
        int *s = "\0";
        if(strcmp(s,NULL)== 0)
        printf("\n s is null");
        else
        printf("\n s is not null");
        }
```

**ans: error**

1002. main()
```c
        {
        int *s = "";
        if(strcmp(s,NULL)== 0)
        printf("\n s is null");
        else
        printf("\n s is not null");
        }
```

**ans: error**

1003. int arr[] = {1,2,3,4}
```c
    int *ptr=arr;
    *(arr+3) = *++ptr + *ptr++;
    Final contents of arr[]
```

**ans: 1,2,3,4**

```
1004. func(int *i, int*j)
          {
          *i=*i * *i;
          *j=*j* *j;
          }

      main()
          {
          int i = 5, j = 2;
          func(&i,&j);
          printf("%d %d", i, j);
          }

          ans: 25 4

1005. int x(char *a)
          {
          a=(char *) malloc(10*sizeof(char));
          *a="hello";
          }

      main()
          {
          char *a="new";
          x(a);
          printf("%s",a);
          }

          ans: error (nonportable pointer conversion)

1006. int x(char *a)
          {
          char *b;
          a=(char *) malloc(10*sizeof(char));
          b=(char *) malloc(10*sizeof(char));
          a="hello";
          b=a;
          }

      main()
          {
          char *a="new";
          x(a);
          printf("%s",a);
          }

          ans: new

1007. int x(char *a)
          {
          char b[10];
```

```
            a=(char *) malloc(10*sizeof(char));
            a="hello";
            b=a;
            }

    main()
            {
            char *a="new";
            x(a);
            printf("%s",a);
            }
```

**ans: error (lvalue required. strcpy should be used)**

1008. a. for(i=0;i<num;i++)
      b. for(i=num;i>0;i--)
      Assuming no code optimization and assume that the
      microprocessor
      has flags etc. which one is correct

          **ans: b (in 'b' zero flag is tested but in 'a'  both
      compare instruction and flag testing will be there)**


1009. will these two work in same manner
      #define intp int *
      typedef int * inpp;

          **ans: no**

      #define intp int *
      typedef int * inpp;
      main()
            {
            inpp t1,t2;
            intp m1,m2;
            printf("%d %d %d
      %d",sizeof(t1),sizeof(t2),sizeof(m1),sizeof(m2));
            }

          **ans: 4 4 4 2 (t1,t2 and m1 are pointers and m2 is
      integer)**

1010. #define max 10
      main()
            {
            int a,b;
            int *p,*q;
            a=10;b=19;
            p=&(a+b);
            q=&max;
```

```
        }
```

**ans: error (& must take address of a memory location)**

1011. main()
```
        {
        char S[6]= "HELLO";
        printf("%s ",S[6]);
        }
```

**ans: error (trying to print from memory location zero)**

1012. unsigned char c;
```
    for ( c=0;c!=256;c++2)
    printf("%d",c);
```

No. of times the loop is executed ?

**ans: infinite times**

1013. main()
```
        {
        char *x="string";
        char y[]="add";
        char *z;
        z=(char *) malloc(sizeof(x)+sizeof(y)+1);
        strcpy(z,y);
        strcat(z,x);
        printf("%s+%s=%s",y,x,z);
        }
```

**ans: add+string=addstring**

1014. char *(*(*a[n]) () )();

**ans:an array of n pointers to functions returning
pointers to functins returning pointers to characters**

1015. What does the following piece of code do ?
```
    sprintf(retbuf, "%d", n);
```

    (A) Print the Integer value of n
    (B) Copy the string representation of the integer variable n
into the buffer retbuf
    (C) Print the Float value of n.
    (D) Print the string representation of the integer variable n.

**ans: (B)**

1016. What is wrong with the program
```
    double d;
```

```
scanf("%f", &d);

(A) Instead of %f , %lf should be used for formatting
(B) Instead of %f , %d should be used for formatting
(C) Instead of %f , %D should be used for formatting
(D) Instead of %f , %n should be used for formatting
```

**ans: (A)**

```
1017. void func()
      {
      int x = 0;
      static int y = 0;
      x++; y++;
      printf( "%d--%d ", x, y );
      }
      int main()
      {
      func();
      func();
      return 0;
      }
```

**ans: 1—1 1—2**

```
1018. main()
      {
      int I,j;
      for(I=0, j=I++; j>I; j++, I++)
      {
      printf("%d %d", I, j);
      }
      }
```

**ans: no output**

```
1019. void main()
      {
      int z;
      int x = 5;
      int y = -10;
      int a = 4;
      int b = 2;
      z = x++ - --y * b /a;
      printf("%d",z);
      }
```

**ans: 10**

```
1020. void main()
      {
```

```
        int x[] = { 1, 4, 8, 5, 1, 4 };
        int *ptr, y;
        ptr = x + 4;
        y = ptr - x;
        printf("%d",y);
        }
```

**ans: 4**

1021. void main()
```
        {
        char str[20] = "ENIGMA";
        char *p, *q, *r;
        p=str;
        q=p++;
        r=p+3 - (p-q);
        printf("%3s %5s", (++p)+3, r);
        }
```

**ans:   A    GMA**

1022. void main()
```
        {
        char str[20] = "ENIGMA";
        char *p, *q, *r;
        p=str;
        q=p++;
        r=p+3 - (q-p);
        printf("%3s %5s", (++p)+3, r);
        }
```

**ans:   A     A**

1023. void inc_count(int count)
```
        {
        count ++;
        }
    int main()
        {
        int count = 0;
        while (count < 10)
        inc_count(count);
        return count ;
        }
```
        What will be the value returned by the function main?

**ans: infinite loop (control will not come to return statement)**

1024. What is the difference between the two declaration ?

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

(A) No Difference
(B) The 2nd declaration will not compile
(C) First case Compiler looks in all default location
and in 2nd case only in the working directory
(D) Depends on the Compiler
```

**ans: (C)**

1025. ```
#define FIRST_PART 7
#define LAST_PART 5
#define ALL_PARTS FIRST_PART + LAST_PART

int main()
    {
    printf ("The Square root of all parts is %d\n" ,
ALL_PARTS * ALL_PARTS);
    return(0);
    }
```

**ans: The Square root of all parts is 47**

1026. ```
void *p;
what operation cannot be performed on p?
```

**ans : arithmetic operation unless it is properly typecasted**

1027. ```
main()
    {
    char **p="Hello";
    printf("%s ",p);
    printf("%c",*p);
    //printf("%c",**p);
    }
```

**ans: Hello H**

1028. ```
main()
    {
    char **p="Hello";
    printf("%s ",p);
    printf("%c",*p);
    printf("%c",**p);
    }
```

**ans: error (trying to access memory location 72 which may not be accessible)**

```
1029. main()
          {
          char str[]="Geneius";
          print (str);
          }
      print(char *s)
          {
          if(*s)
          print(++s);
          printf("%c ",*s);
          }

          ans: null null s u i e n e (null means null character)

1030. main()
          {
          printf("Genius %d",fun(123));
          }
      fun(int n)
          {
          return (printf("%d",n));
          }

          ans: 123Genius 3

1031. main()
          {
          int i=4;
          fun(i=i/4);
          printf("%d",i);
          }
      fun(int i)
          {
          return i/2;
          }

          ans: 1

1032. main()
          {
          printf("\"NITK %%SURATHKAL%% !\"");
          }

          ans: "NITK %SURATHKAL% !"

1033. main()
          {
          printf("\"NITK \%SURATHKAL\% !\"");
          }
```

ans: "NITK %SURATHKAL% !"

1034. main()
```
{
char str[7]="strings";
printf("%s",str);
}
```

**ans: strings……….(till it encounters null character.
While printing if it accesses inaccessible memory
location error will come)**

1035. main()
```
{
char str[8]="strings";
printf("%s",str);
}
```

**ans: strings**

1036. main()
```
{
char *p = "Oracle India";
p[5] == 'l' ? printf("Orcle") : printf("India");
}
```

**ans: India**

1037. main()
```
{
int i=5;
recursive(i);
}
recursive(int u)
{
if(u > 0 )
recursive(u-1);
printf("%d ", u);
}
```

**ans: 0 1 2 3 4 5**

1038. char *(*(*x() ) [] ) ()

**ans: x is a function returnting pointer to array of
pointers to functions returning character pointers**

1039. const int MAX=10;
```
main()
{
enum a {a,b,MAX};
```

```
        printf("%d",MAX);
        }

        ans: 2
```

1040. main()
```
        {
        const int MAX=10;
        enum a {a,b,MAX};
        printf("%d",MAX);
        }

        ans: error (multiple declaration of MAX)
```

1041. const int MAX=10;
```
    main()
        {
        enum a {a,b,MAX};
        MAX=3;
        printf("%d",MAX);
        }

        ans: error (lvalue required)
```

1042. 1)enum object is a const which can only be assigned a value
     at  initialization or 2) a variable which can be assigned any
     value in the middle of the program?

        **ans: 1) is correct**

1043. void *p;
     what operation cannot be performed on p?

        **ans : arithmetic operation unless it is properly
    typecasted**

1044. main()
```
        {
        int i=4;
        fun(i=i/4);
        printf("%d",i);
        }
    fun(int i)
        {
        return i/2;
        }
```

        **ans: 1**

1045. main()
```
        {
```

```
        int a=500,b,c;
        if(a>400)
        b=300; c=2--; printf("%d %d",b,c);
        }

        ans: error (lvalue required)
```

1046. main()
```
        {
        char c1='a',c2='Z';
        if (c1=='a'or c2=='z')
        printf("welcome");
        }

        ans: error (for ORing || symbol should be used)
```

1047. main()
```
        {
        int i;
        for(i=0;i<=10;i++);
        printf("%d ",i);
        }

        ans: 11
```

1048. main()
```
        {
        int x=10,y,z;
        y=--x;
        z=x--;
        printf("%d %d %d",x,y,z);
        }

        ans: 8 9 9
```

1049. main()
```
        {
        int i;
        int marks[]={100,90,75,90,80};
        for (i=0;i<4;i++)
        disp(&marks[i]);
        }
    disp(int *n)
        {
        printf("%d ",*n);
        }

        ans: 100 90 75 90
```

1050. main()
```
        {
```

```
int arr[]={1,2,3,4,5,6,7};
int *I,*j;
I=&arr[1];
j=&arr[5];
printf("%d %d",*j+*I,*j-*I);
}
```

**ans: 8 4 (be careful about upper case and lower case)**

1051. main()
```
{
int n=2,sum=5;
switch(n)
{
case 2:sum=sum-2;
case 3:sum*=5;
break;
default:sum=0;
}
printf("%d",sum);
}
```

**ans: 15**

1052. main()
```
{
int i=0;
for(i=0;i<20;i++)
{
switch(i)
{
case 0:
i+=5;
case 1:
i+=2;
case 5:
i+=5;
default:
i+=4;
break;
}
printf("%d ",i);
}
}
```

**ans: 16 21**

1053. main()
```
{
int i=0;
for(i=0;i<20;i++)
```

```
            {
            switch(i)
            {
            default:
            i+=4;
            break;
            case 0:
            i+=5;
            case 1:
            i+=2;
            case 5:
            i+=5;
            }
            printf("%d ",i);
            }
            }
```

**ans: 12 17 22**

1054. main()
```
            {
            int i=0;
            for(i=0;i<20;i++)
            {
            switch(i)
            {
            default:
            i+=4;
            case 0:
            i+=5;
            case 1:
            i+=2;
            case 5:
            i+=5;
            }
            printf("%d ",i);
            }
            }
```

**ans: 12 29**

1055. func(int i)
```
            {
            if(i%2) return 0;
            else return 1;
            }
     main()
            {
            int i=3;
            i=func(i);
            i=func(i);
```

```
        printf("%d",i);
        }

        ans: 1
```

1056. char*g()
```
        {
        static char x[1024];
        return x;
        }
    main()
        {
        char*g1="First String";
        strcpy(g(),g1);
        g1=g();
        strcpy(g1,"Second String");
        printf("Answer is:%s", g());
        }

        ans: Answer is:Second String
```

1057. main()
```
        {
        int a[5]={1,3,6,7,0};
        int *b;
        b=&a[2];
        printf("%d",b[-1]);
        }

        ans: 3
```

1058. Given a piece of code
```
    int x[10];
    int *ab;
    ab=x;
    To access the 6th element of the array which of the following
is incorrect?
    (A)   *(x+5) (B) x[5] (C) ab[5] (D) *(*ab+5}

        ans: (D)
```

1059. main()
```
        {
        int i = 5;
        printf("%d\n", i--*i++);
        }

        ans: 20
```

1060. main()
```
        {
```

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

**ans: 30**

1061. main()
```
        {
        int i = 5;
        printf("%d %d", i,i++*i--*i++);
        }
```

**ans: 6 150**

1062. main()
```
        {
        char ch='a';
        printf("%d ",ch);
        printf("%d",((int)ch)++);
        }
```

**ans: error (lvalue required)**

1063. int main()
```
        {
        int i;
        int array1[10], array2[10]={1,2,3,4,5,6,7,8,9,10};
        int *ep, *ip2 = &array2[0];
        int *ip1 = &array1[0];
        for(ep = &array1[9]; ep >= ip1; ep--)
        *ep = *ip2++ ;
        for(i=0;i<10;i++)
        printf("%d ",array1[i]);
        }
```

**ans: copies array2 to array1 in reverse order (10 9 8 7 6 5 4 3 2 1)**

1064. int main()
```
        {
        char string[100];
        char *p;
        gets(string);
        for(p = string; *p != '\0'; p++);
        printf("%d", p - string);
        }
```

**ans: prints the length of "string"**

1065. main()
```
        {
```

```
        int i=1 ;
        for (;;);
        {
        if(i==1)
        {
        printf("%d",i);
        exit();
        }
        }
        }
```

**ans: infinite loop (no output)**

1066. ```
const int n = 7;
int a[n];
main()
        {
        ;
        }
```

**ans: error (constant expression required for array size)**

1067. ```
void main()
        {
        char *p ;
        p = (char*)malloc(100);
        strcpy(p,"Oracle India");
        (p[5] == 'l') ? printf("Oracle") : printf("India");
        }
```

**ans: India**

1068. ```
void main()
        {
        int a=5,b,i;
        int func(int y);
        for(i = 0;i < 5;i++)
        {
        a = b = func(a);
        printf("%d ",b);
        }
        }

int func(int y)
        {
        static int x = 0;
        x++;
        y = y + x;
        return(y);
        }
```

**ans: 6 8 11 15 20**

1069. void main()
```
{
char i;
for(i=0;i<=256;i++)
printf("%d",i);
}
```

**ans: infinite loop**

1070. void main()
```
{
int ret,I = 10;
ret = func1(I);
printf("%d",ret);
}
```

```
int func1(int d)
{
int ret1;
ret1 = func2(--d);
return(ret1);
}
```

```
int func2(int y)
{
return(++y);
}
```

**ans: 10 (replace ––d with d––    then answer will be 11)**

1071. void main()
```
{
char str[20];
strcpy(str,"Oracle India");
printf("%c",str[10]);
}
```

**ans: i**

1072. void main()
```
{
int I=0,j=1;
printf("%d %d",--I ,j++);
}
```

**ans: -1 1**

1073. .#define sq(a) (a*a)

```
printf ("%d",sq (3+2));
```

**ans: 11**

1074. #define max 20
```
printf ("%d", ++max);
```

**ans: lvalue required (error)**

1075. Which of the following 'return' statement is correct?

```
return, return;
return(1, 2, 3);
return(return 4);
(return 5, return 6);
```

**ans: return (1,2,3) is correct and 3 will be returned**

1076. void main()
```
{
char buffer[10] = {"Genesis"};
printf(" %d ", &buffer[4]- (buffer));
}
```

**ans: 4**

1077. void main()
```
{
struct a
{
char ch[10];
char *str;
};
struct a s1={"Hyderabad","Bangalore"};
printf("\n%c%c ",s1.ch[0],*s1.str);
printf("%s %s",s1.ch,s1.str);
getch();
}
```

**ans: HB Hyderabad Bangalore**

1078. void main()
```
{
int i,j,k;
for(i=0;i<3;i++)
k=sum(i,i);
printf("\n%d",k);
getch();
}
```

```
sum(s,t)
        {
        static int m;
        m+=s+t;
        return m;
        }
```

**ans: 6**

1079. void main()
```
        {
        int i;
        for(i=1;i<6;++i)
        switch(i)
        {
        case 1:
        case 2: printf("%d,",i++);break;
        case 3: continue;
        case 4: printf("%d,",i);
        }
        printf("%d",i);
        getch();
        }
```

**ans: 1,4,6**

1080. void main()
```
        {
        char s[]="oracle is the best";
        char t[40];
        char *ss,*tt;
        while(*tt++=*ss++);
        printf("%s",t);
        getch();
        }
```

**ans: core dump (Garbage value)**

1081. void main()
```
        {
        int j[10]={9,7,5,3,1,2,4,6,9};
        int i=1;
        clrscr();
        for(;i<9;i++)
        printf("%d ",--j[i++]);
        getch();
        }
```

**ans: 6 2 1 5**

1082. void main()

```
{
int i,j,k,n=5;
clrscr();
for(i=5;i>0;i--)
{
j=1<i;
k=n&j;
k==0?printf("0"):printf("1");
}
getch();
}
```

**ans: 11110**

1083. union
```
{
int a;
char b;
char c[10];
}u1;
void main()
{
int l=sizeof(u1);
printf("%d",l);
getch();
}
```

**ans: 10**

1084. void main()
```
{
struct a
{
int i;
char *st1;
};
typedef struct a ST;
ST *str1;
str1=(ST*)malloc(100);
str1->i=100;
strcpy(str1->st1,"Welcome to Oracle");
printf(" %d %s\n",str1->i,str1->st1);
getch();
}
```

**ans: 100 Welcome to Oracle**

1085. void main()
```
{
int i,j,k;
i=2;
```

```
        j=4;
        k=i++>j&2;
        printf("%d",k);
        if(++k && ++i<--j|| i++)
        {
        j=++k;
        }
        printf(" %d %d %d",i,-j--,k);
        getch();
        }

        ans: 0 -5 -2 2
```

1086. Which of the following is not true incase of
      Command line arguments

      A.The argc parameter is used to hold the number
      of arguments in the =
      command line and is an integer
      B. The argv parameter is a pointer to an array of
      a character =
      pointer and each one points to command line
      arguments
      C. The argv[1] always point to program name
      D. None of above

        ans: C

1087. void main()
```
        {
        int i,j=20;
        clrscr();
        for(i=1;i<3;i++)
        {
        printf("%d,",i);
        continue;
        printf("%d",j);
        break;
        }
        getch();
        }
```
        ans: 1,2,

1088. void fn(int *a, int *b)
```
        {
        int *t;
        t=a;
        a=b;
        b=t;
        }
```

```
main()
    {
    int a=2;
    int b=3;
    fn(&a,&b);
    printf("%d %d",a,b);
    }

    ans: 2 3
```

1089.
```
main()
    {
    char *p="abc";
    char *q="abc123";
    while(*p=*q)
    {
    printf("%c %c ",*p,*q);
    getch();
    }
    }

    ans: a a a a a a a a a a a….(infinite loop)
```

1090.
```
#define void int
int i=300;
void main(void)
    {
    int i=200;
    {
    int i=100;
    printf("%d ",i);
    }
    printf("%d",i);
    }

    ans: error
```

1091.
```
#define void int
int i=300;
void main(void argc)
    {
    int i=200;
    {
    int i=100;
    printf("%d ",i);
    }
    printf("%d",i);
    }

    ans: 100 200
```

```
1092. main()
      {
      int A=5,x;
      int fun(int *, int);
      x=fun(&A,A);
      printf("%d",x);
      }

      int fun(int *x, int y);
      {
      *x=*x+1;
      return(*x*y);
      }

      ans: error (; in function definition)

1093. main()
      {
      int A=5,x;
      int fun(int *, int);
      x=fun(&A,A);
      printf("%d",x);
      }

      int fun(int *x, int y);

      ans: linker error (undefined symbol  fun)

1094. main()
      {
      int A=5,x;
      int fun(int *, int);
      x=fun(&A,A);
      printf("%d",x);
      }

      int fun(int *x, int y)
      {
      *x=*x+1;
      return(*x*y);
      }

      ans: 30

1095. main()
      {
      int i;
      int x[]={0,0,0,0,0};
      for(i=1;i<=4;i++)
      x[x[i]]++;
      for(i=0;i<5;i++)
```

```
        printf(" %d",x[i]);
        }

        ans: 4 0 0 0 0
```

1096. main()
```
        {
        int i,j,count;
        int a[3][4] = { -1,2,3,-4,5,6,7,-8,9,10,11,12};
        count=0;
        for(i=2;i<1;i--)
        {
        for(j=3;j<1;j--)
        {
        if(a[i][j]<1)
        count+=1;
        }
        }
        printf("%d",count);
        }

        ans: 0
```

1097. int sum,count;
```
     void main(void)
        {
        for(count=5;sum+=--count;)
        printf("%d ",sum);
        }

        ans: 4 7 9 10 10 9 7 4
```

1098. void main(void)
```
        {
        int i;
        for(i=2;i<=7;i++)
        printf("%5d",fno());
        }
     fno()
        {
        static int f1=1,f2=1,f3;
        return(f3=f1+f2,f1=f2,f2=f3);
        }

        ans: 2 3 5 8 13 21
```

1099. void main (void)
```
        {
        int x;
        x = 0;
        if (x=0)
```

```
        printf ("Value of x is 0");
        else
        printf ("Value of x is not 0");
        }

        **ans: Value of x is not 0**

1100. int foo(char *);
        void main (void)
        {
        char arr[100] = {"Welcome to Mistral"};
        foo (arr);
        }
    foo (char *x)
        {
        printf ("%d\t",strlen (x));
        printf ("%d\t",sizeof(x));
        return 0;
        }

        **ans: 18 4**

1101. display()
        {
        printf (" Hello World");
        return 0;
        }
    void main (void)
        {
        int (*func_ptr)();
        func_ptr = display;
        (* func_ptr)();
        }

        **ans: Hello World**

1102. void main (void)
        {
        int i=0;
        char ch = 'A';
        do
        putchar (ch);
        while(i++ < 5 || ++ch <= 'F');
        printf("%c ",ch);
        }

        **ans: AAAAAABCDEFG**

1103. char *rev();
    void main(void)
        {
```

```
printf ("%c", *rev());
}
char *rev ()
{
char dec[]="abcde";
return dec;
}
```

**ans: a (another ans: prints garbage, address of the local variable should not returned)**

1104. void main(void)
```
{
int i;
static int k;
if(k=='0')
printf("one");
else if(k== 48)
printf("two");
else
printf("three");
}
```

**ans: three**

1105. void main(void)
```
{
enum sub{chemistry, maths, physics};
struct result
{
char name[30];
enum sub sc;
};
struct result my_res;
strcpy (my_res.name,"Patrick");
my_res.sc=physics;
printf("name: %s   ",my_res.name);
printf("pass in subject: %d\n",my_res.sc);
}
```

**ans: name: Patrick   pass in subject: 2**

1106. main()
```
{
char *p = "MISTRAL";
printf ("%c\t", *(++p));
p -=1;
printf ("%c\t", *(p++));
}
```

**ans: I      M**

1107. What does the declaration do?
```
int (*mist) (void *, void *);
```

**ans: declares mist as a pointer to a function that has two void * arguments and returns an int.**

1108. void main (void)
```
{
int mat [5][5],i,j;
int *p;
p = & mat [0][0];
for (i=0;i<5;i++)
for (j=0;j<5;j++)
mat[i][j] = i+j;
printf ("%d\t", sizeof(mat));
i=4;j=5;
printf( "%d", *(p+i+j));
}
```

**ans: 100    5**

1109. void main (void)
```
{
char *p = "Bangalore";
#if 0
printf ("%s", p);
#endif
}
```

**ans: no output**

1110. void main (void)
```
{
char *p = "Bangalore";
#if 1
printf ("%s", p);
#endif
}
```

**ans: Bangalore**

1111. main()
```
{
int x;
float y;
y = *(float *)&x;
}
```

**ans: the program containing the expression compiles and runs without any errors**

```
1112. int main()
           {
           char *a= "Novell";
           char *b;
           b=malloc(10*sizeof(char));
           memset(b,0,10);
           while(*b++=*a++);
           printf("%s",b);
           getch();
           return 0;
           }
```

**ans: no output**

1113. int *(*p[10])(char *)

**ans: array of pointers to functions with character
pointer as argument and returning pointer to integer**

```
1114. main()
            {
             printf("hello"):
             main();
            }
```

**ans: hellohello….(prints recursively till stack
overflows)**

```
1115. #define scanf "%s is a string"
      main()
            {
            printf(scanf,scanf);
            }
```

**ans: %s is a string is a string**

```
1116. main()
            {
            printf("%u",-1);
            }
```

**ans: 65535**

1117. automatic variables are destroyed after function ends because

```
      a)stored in swap
      b)stored in stack and poped out after function returns
      c)stored in data area
      d)stored in disk
```

**ans: b)**

1118. main()
```
{
printf(5+"facsimile");
}
```

**ans: mile**

1119. How to fine the size of the int without using size of operator?

**ans. store -1 in that location so by two's complement all ones will be stored in that location. Keep right shifting it so zeros will be appened on the left. Once the location is filled with all zeros, the number of shifts gives you the size of that operator.**

1120. main()
```
{
char a[2];
*a[0]=7;
*a[1]=5;
printf("%d",&a[1]-a);
}
```

**ans: error (invalid indirection)**

1121. main(){
```
char a[]="hellow";
char *b="hellow";
char c[5]="hellow";
printf("%s %s %s ",a,b,c);
printf(" ",sizeof(a),sizeof(b),sizeof(c));
}
```

**ans: error (too many initializers)**

1122. main()
```
{
float value=10.00;
printf("%g %0.2g %0.4g %f",value,value,value,value);
}
```

**ans: 10 10 10 10.000000**

1123. Which one has no L-Value

```
[i] a[i]
[ii] i
[iii] 2
```

```
     [iv] *(a+i)

             ans. [iii]

1124. main()
          {
          int i=10,j;
          for(j=0;j<1;j++)
          {
          int i=20;
          printf("%d ",i);
          }
          printf("%d",i);
          }

          ans: 20 10

1125. main()
          {
          int i;
          printf("%d",i);
          }
          extern int i=20;

          ans: garbage value

1126. main()
          {
          extern int i;
          printf("%d",i);
          }
          int i=20;

          ans: 20

1127. main()
          {
          int n=6;
          printf("%d",n)
          ;
          }

          ans: 6

1128. main()
          {
          int arr[5]={2,4};
          printf("%d %d %d \n",arr[2],arr[3],arr[4]);
          }

          ans: 0 0 0
```

1129. main()
```
    {
    struct e
    {
    char name[20];
    int a;
    float b;
    };
    struct e ast={"Hell"};
    printf("%d %f \n",ast.a,ast.b);
    }
```

**ans: 0 0.000000**

1130. Given an array of size N in which every number is between 1 and N, determine if there are any duplicates in it. You are allowed to destroy the array if you like.

**ans: 1)compare all the elements with the selected element 2)put it in ascending order and compare adjacent elements**

1131. Given an array of characters which form a sentence of words, give an efficient algorithm to reverse the order of the words (not characters) in it.

**ans: take an array of pointers and  and chage the addresses of the pointers**

1132. test whether a number is a power of 2.

**ans: first test whether it is even or odd and the bitcount. If bitcount is one it is a power of 2.**

1133. Given two strings S1 and S2. Delete from S2 all those characters which occur in S1 also and finally create a clean S2 with the relevant characters deleted.

1134. Reverse a linked list.

**ans: Possible answers —**

```
iterative loop
curr->next = prev;
prev = curr;
curr = next;
next = curr->next
endloop

recursive reverse(ptr)
```

```
        if (ptr->next == NULL)
        return ptr;
        temp = reverse(ptr->next);
        temp->next = ptr;
        return ptr;
        end
```

1135. Given an array t[100] which contains numbers between 1..99.
       Return the duplicated value. Try both O(n) and O(n-square).

1136. Given an array of characters. How would you reverse it. ? How
       would you reverse it without using indexing in the array.

          **ans: use pointers**

1137. Write, efficient code for extracting unique elements from a
       sorted list of array. e.g. (1, 1, 3, 3, 3, 5, 5, 5, 9, 9, 9,
       9) -> (1, 3, 5, 9).

1138. Given an array of integers, find the contiguous sub-array with
       the largest sum.

1139. An array of integers. The sum of the array is known not to
       overflow an integer. Compute the sum. What if we know that
       integers are in 2's complement form?

       ans: If numbers are in 2's complement, an ordinary looking
       loop like
       for(i=total=0;i< n;total+=array[i++]); will do. No need to
       check for overflows!

1140. Write a program to remove duplicates from a sorted array.

       ans:  int remove_duplicates(int * p, int size)
       {
       int current, insert = 1;
       for (current=1; current < size; current++)
       if (p[current] != p[insert-1])
       {
       p[insert] = p[current];
       current++;
       insert++;
       } else
       current++;
       return insert;
       }

1141. Write an efficient C code for 'tr' program. 'tr' has two
       command line arguments. They both are strings of same length.
       tr reads an input file, replaces each character in the first
```

string with the corresponding character in the second string.
eg. 'tr abc xyz' replaces all 'a's by 'x's, 'b's by 'y's and
so on. ANS.
a) have an array of length 26.
put 'x' in array element corr to 'a'
put 'y' in array element corr to 'b'
put 'z' in array element corr to 'c'
put 'd' in array element corr to 'd'
put 'e' in array element corr to 'e'
and so on.

the code
```
while (!eof)
{
c = getc();
putc(array[c - 'a']);
}
```

1142. Write a program to find whether a given m/c is big-endian or
little-endian!

1143. If you're familiar with the ? operator x ? y : z
you want to implement that in a function: int cond(int x, int
y, int z); using only ~, !, ^, &, +, |, <<, >> no if
statements, or loops or anything else, just those operators,
and the function should correctly return y or z based on the
value of x. You may use constants, but only 8 bit constants.
You can cast all you want. You're not supposed to use extra
variables, but in the end, it won't really matter, using vars
just makes things cleaner. You should be able to reduce your
solution to a single line in the end though that requires no
extra vars.

1144. Under what circumstances can one delete an element from a
singly linked list in constant time?

ans: If the list is circular and there are no references to
the nodes in the list from anywhere else! Just copy the
contents of the next node and delete the next node. If the
list is not circular, we can delete any but the last node
using this idea. In that case, mark the last node as dummy!

1145. Given a singly linked list, determine whether it contains a
loop or not.

ans: (a) Start reversing the list. If you reach the head,
gotcha! there is a loop!
But this changes the list. So, reverse the list again.
(b) Maintain two pointers, initially pointing to the head.
Advance one of them one node at a time. And the other one, two

nodes at a time. If the latter overtakes the former at any
time, there is a loop!

```
            p1 = p2 = head;

            do {
                p1 = p1->next;
                p2 = p2->next->next;
                    } while (p1 != p2);
```

1146. Given a singly linked list, print out its contents in reverse
      order. Can you do it without using any extra space?

      ans: Start reversing the list. Do this again, printing the
contents.

1147. Reverse a singly linked list recursively. function prototype
      is node * reverse (node *) ;

      ans:
       node * reverse (node * n)
          {
                node * m ;

                if (! (n && n -> next))
                  return n ;

                m = reverse (n -> next) ;
                n -> next -> next = n ;
                n -> next = NULL ;
                return m ;
          }

1148. Given a singly linked list, find the middle of the list.

      HINT. Use the single and double pointer jumping. Maintain two
      pointers, initially pointing to the head. Advance one of them
      one node at a time. And the other one, two nodes at a time.
      When the double reaches the end, the single is in the middle.
      This is not asymptotically faster but seems to take less steps
      than going through the list twice.

1149. Reverse the bits of an unsigned integer.

      ans:

       #define reverse(x)                                    \
                    (x=x>>16|(0x0000ffff&x)<<16,             \
                        x=(0xff00ff00&x)>>8|
(0x00ff00ff&x)<<8, \
```

```
                              x=(0xf0f0f0f0&x)>>4|
(0x0f0f0f0f&x)<<4, \
                              x=(0xcccccccc&x)>>2|
(0x33333333&x)<<2, \
                              x=(0xaaaaaaaa&x)>>1|(0x55555555&x)<<1)
```

1150. Compute the number of ones in an unsigned integer.

    ans:

```
#define count_ones(x)                           \
                (x=(0xaaaaaaaa&x)>>1+(0x55555555&x), \
                 x=(0xcccccccc&x)>>2+(0x33333333&x), \
                 x=(0xf0f0f0f0&x)>>4+(0x0f0f0f0f&x), \
                 x=(0xff00ff00&x)>>8+(0x00ff00ff&x), \
                 x=x>>16+(0x0000ffff&x))
```

1151. Compute the discrete log of an unsigned integer.

    ans:

```
#define discrete_log(h) \
 (h=(h>>1)|(h>>2), \
 h|=(h>>2), \
 h|=(h>>4), \
 h|=(h>>8), \
 h|=(h>>16), \
 h=(0xaaaaaaaa&h)>>1+(0x55555555&h), \
 h=(0xcccccccc&h)>>2+(0x33333333&h), \
 h=(0xf0f0f0f0&h)>>4+(0x0f0f0f0f&h), \
 h=(0xff00ff00&h)>>8+(0x00ff00ff&h), \
 h=(h>>16)+(0x0000ffff&h))
```

If I understand it right, log2(2) =1, log2(3)=1,
log2(4)=2..... But this macro does not work out log2(0) which
does not exist! How do you think it should be handled?

1152. How do we test most simply if an unsigned integer is a power
of two?

        ans: `#define power_of_two(x) \ ((x)&&(~(x&(x-1))))`

1153. Set the highest significant bit of an unsigned integer to
zero.

    ans: Set the highest significant bit of an unsigned integer to
zero

```
#define zero_most_significant(h) \
(h&=(h>>1)|(h>>2), \
h|=(h>>2), \
```

```
h|=(h>>4), \
h|=(h>>8), \
h|=(h>>16))
```

1154. You're given an array containing both positive and negative
      integers and required to find the sub-array with the largest
      sum (O(N) a la KBL). Write a routine in C for the above.

1155. Given two strings S1 and S2. Delete from S2 all those
      characters which occur in S1 also and finally create a clean
      S2 with the relevant characters deleted.

1156. Besides communication cost, what is the other source of
      inefficiency in RPC? (answer : context switches, excessive
      buffer copying). How can you optimize the communication?
      (ans : communicate through shared memory on same machine,
      bypassing the kernel _ A Univ. of Wash. thesis)

1157. An array of characters. Reverse the order of words in it.

      ans: Write a routine to reverse a character array. Now call it
      for the given array and for each word in it.

1158. Given a list of numbers ( fixed list) Now given any other
      list, how can you efficiently find out if there is any element
      in the second list that is an element of the first list (fixed
      list).

1159. Print an integer using only putchar. Try doing it without
      using extra storage.

1160. int *a;
      char *c;
      *(a) = 20;
      *c = *a;
      printf("%c",*c);

      what is the output?

      Before using pointer they should be assigned some address

1161. to reverse a string using a recursive function, without
      swapping or using an extra memory.

1162. Give the outputs of a compiler and assembler and loader and
      linker etc.

1163. Tell about strtok & strstr functions.

1164. #define int sizeof(int)
      main()
```

```
        {
        printf("%d",int);
        }

        ans: 2

1165. #define i sizeof(i)
    main()
        {
        printf("%d",i);
        }

        ans: error (undefined symbol i)
```