2)POINTER QUIZ

1) What is the output of above program?

```
#include<stdio.h>
int main()
{
    int a;
    char *x;
    x = (char *) &a;
    a = 512;
    x[0] = 1;
    x[1] = 2;
    printf("%dn",a);
    return 0;
}
```

- ◯ A) Machine dependent
- ◯ B) 513
- ◯ C) 258
- ◯ D) Compiler Error

OUTPUT: Machine dependent.
EXPLANATION: Output is 513 in a little endian machine. To understand this output, let integers be stored using 16 bits. In a little endian machine, when we do x[0] = 1 and x[1] = 2, the number a is changed to 00000001 00000010 which is representation of 513 in a little endian

……

2) Assume that an int variable takes 4 bytes and a char variable takes 1 byte.

```
#include<stdio.h>
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    int *ptr1 = arr;
    int *ptr2 = arr + 5;
    printf("Number of elements between two pointer are: %d.",
                                    (ptr2 - ptr1));
    printf("Number of bytes between two pointers are: %d",
                                  (char*)ptr2 - (char*) ptr1);
    return 0;
}
```

- ◯ (A) Number of elements between two pointer are: 5.Number of bytes between two pointers are: 20
- ◯ (B) Number of elements between two pointer are: 20.Number of bytes between two pointers are: 20
- ◯ (C) Number of elements between two pointer are: 5.Number of bytes between two pointers are: 5
- ◯ (D) Compiler Error
- ◯ (E) Runtime Error

OUTPUT: A
EXPLANATION: Array name gives the address of first element in array. So when we do '*ptr1 = arr;', ptr1 starts holding the address of element 10. 'arr + 5' gives the address of 6th element as arithmetic is done using pointers. So 'ptr2-ptr1' gives 5. When we do '(char *)ptr2', ptr2 is type-casted to char pointer and size of character is one byte, pointer arithmetic happens considering character pointers. So we get 5*sizeof(int)/sizeof(char) as a difference of two pointers.

……

3)What will the output of the program?

```c
#include<stdio.h>

void fun(void *p);
int i;

int main()
{
    void *vptr;
    vptr = &i;
    fun(vptr);
    return 0;
}
void fun(void *p)
{
    int **q;
    q = (int**)&p;
    printf("%d\n", **q);
}
```

- A. Error: cannot convert from void** to int**
- B. Garbage value
- C. 0
- D. No output

OUTPUT:C(0)
EXPLANATION: first we define uninitialized global variable, so the compiler initialize it to zero, and in main we define general pointer and assign to it address of global variable the pass this pointer to function, in fun we define pointer to pointer and casting the general pointer with pointer to pointer and print the value that point to global variable.

……

4)Point out the error in the program?

```c
#include<stdio.h>

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

- [A]. Error: Declaration syntax
- [B]. Error: Expression syntax
- [C]. Error: LValue (left hand side) required
- [D]. Error: Rvalue (right hand side) required

OUTPUT: C
EXPLANATION: we can’t use pre or post increment with the name of array(name of array is constant pointer).

5)what will be output when you will execute following c code?

```
#include<stdio.h>
void main(){
    int array[2][3]={5,10,15,20,25,30};
    int (*ptr)[2][3]=&array;
    printf("%d\t",***ptr);
    printf("%d\t",***(ptr+1));
    printf("%d\t",**(*ptr+1));
    printf("%d\t",*(*(*ptr+1)+2));
}
```

- (A) 5 Garbage 20 30
- (B) 10 15 30 20
- (C) 5 15 20 30
- (D) Compilation error
- (E) None of the above

OUTPUT: A
EXPLANATION: First we define two dimensional array and then we define pointer to two dimensional array and points to the whole array, so when we print the first line, we print the first element in array the increment pointer by one so it points to another location so will print garbage and third line will print fourth element in array(**(*ptr+1) is equal to *(*(*ptr+1)+0) is equal to array[1][0]) and the last line will print array[1][2].

……

6)Consider the following variable declarations and definitions.

```
i) int var_9 = 1;
ii) int 9_var = 2;
iii) int _ = 3;
```

- Both i) and iii) are valid.
- Only i) is valid.
- Both i) and ii) are valid.
- All are valid.

OUTPUT: Both i) and iii) are valid.
EXPLANATION: we can't start name of variable with number.

7) The following 'C' statement : int * f [ ] ( ); declares:

- A function returning a pointer to an array of integers.
- Array of functions returning pointers to integers.
- A function returning an array of pointers to integers.
- An illegal statement.

OUTPUT: Array of functions returning pointers to integers
EXPLANATION: A function returning a pointer to an array of integers-> int *(*function())[4];
A function returning an array of pointers to integers-> int (*p[])();

……

8)what will be the output of the program?

```c
#include<stdio.h>
power(int**);
int main()
{
    int a=5, *aa; /* Address of 'a' is 1000 */
    aa = &a;
    a = power(&aa);
    printf("%d\n", a);
    return 0;
}
power(int **ptr)
{
    int b;
    b = **ptr***ptr;
    return (b);
}
```

- [A]. 5
- [B]. 25
- [C]. 125
- [D]. Garbage value

OUTPUT: B(25)
EXPLANATION: we define variable and pointer points to this variable and we pass address of pointer to function and receive it in pointer to pointer and in the fun we define variable and assign to it multiple of variable a.

9) Are the expression *ptr++ and ++*ptr are same?

- ○ True
- ○ False

OUTPUT: False
EXPLANATION: int a=10;
int *p=&a;
*ptr++; //print garbage value
++*ptr; //print 11.

……

10)what will be the output of the program?

```c
#include<stdio.h>

int main()
{
    printf("%c\n", 7["IndiaBIX"]);
    return 0;
}
```

- ○ [A]. Error: in printf
- ○ [B]. Nothing will print
- ○ [C]. print "X"
- ○ [D]. print "7"

OUTPUT:C(print ‘x’).
EXPLANATION: we want to print the seventh element in the array(indiabix).

11) What will be the output of the program assuming that the array begins at the location 1002 and size of an integer is 4 bytes?

```
#include<stdio.h>

int main()
{
    int a[3][4] = { 1, 11, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
    printf("%u, %u, %u\n", a[0]+1, *(a[0]+1), *(*(a+0)+1));
    return 0;
}
```

- A. 448, 4, 4
- B. 520, 2, 2
- C. 1006, 2, 2
- D. Error
- E. 1006,11,11

OUTPUT: E(1006,11,11).
EXPLANATION: The base address of array at location 1002 and then increment pointer by one and size of pointer is 4 bytes so a[0]+1 will print 1006 and the second and third at the same-> a[0][1].

.....

12) In the following program add a statement in the function `fun()` such that address of `a` gets stored in `j`?

```
#include<stdio.h>
int main()
{
    int *j;
    void fun(int**);
    fun(&j);
    return 0;
}
void fun(int **k)
{
    int a=10;
    /* Add a statement here */
}
```

- A. **k=a;
- B. k=&a;
- C. *k=&a
- D. &k=*a

OUTPUT: C(*k=&a).

13)what will be the output if you will be compile and execute the following c code?

```
int * call();
void main() {
int *ptr;
ptr=call();
      clrscr();
printf("%d",*ptr);
}
int * call() {
int a=25;
a++;
return &a;
}
```

- A) 25
- (B) 26
- (C) Any address
- (D) Garbage value
- (E) Compiler error

OUTPUT: D(Garbage Value).
EXPLANATION: variable a is local variable, so when the function ends the variable will delete from the memory(variable must be static to store its value).

……

14)what will be the output of the program?

```
#include<stdio.h>

int main()
{
    int i=3, *j, k;
    j = &i;
    printf("%d\n", i**j*i+*j);
    return 0;
}
```

- A. 30
- B. 27
- C. 9
- D. 3

OUTPUT: A(30).
EXPLANATION: we define variable i and pointer points to this variable and then print it 3*3*3+3=30.

15) what will be the output of the program?

```
void main()
{
     struct bitfield
     {
          unsigned a:5;
          unsigned c:5;
          unsigned b:6;

     }bit;
     char *p;
     struct bitfield *ptr,bit1={1,3,3};
     p=&bit1;
     p++;
     clrscr();
     printf("%d",*p);
     getch();
}
```

OUTPUT: 12.
EXPLANATION: Memory is byte addressable so pointer points to byte not bit. Structre bit1 will store in memory like this:10000 11000 110000. So we increment pointer and pointer now points to next byte which is(00110000)->12

……

16) what will be the output of the program?

```
#include<stdio.h>

int main()
{
    char str[] = "peace";
    char *s = str;
    printf("%s\n", s++ +3);
    return 0;
}
```

- ◯ [A]. peace
- ◯ [B]. eace
- ◯ [C]. ace
- ◯ [D]. ce
- ◯ [E]. e

OUTPUT: D(ce).
EXPLANATION: we define pointer points to the first element in array and increment it by 3, so it's points to the third element in array so print from letter c to end.

17) what will be the output of the program?

```c
#include<stdio.h>
int check (int, int);

int main()
{
    int c;
    c = check(10, 20);
    printf("c=%d\n", c);
    return 0;
}
int check(int i, int j)
{
    int *p, *q;
    p=&i;
    q=&j;
    i>=45 ? return(*p): return(*q);
}
```

- A. Print 10
- B. Print 20
- C. Print 1
- D. Compile error

OUTPUT: D(compiler error).
EXPLANATION: We can't use return in the ternary operator. to fix this issue use this line
`return (i>=45) ? (*p): (*q);`
the syntax of a ternary operator is expr1?expr2:expr2;
where exp1,exp2,exp3 are expression and return statement is not an expression

……

18) Which of the following option is correct?

Consider following two C - program :
P1 :

```c
int main()
{
    int (*ptr)(int ) = fun;
    (*ptr)(3);
    return 0;
}

int fun(int n)
{
  for(;n > 0; n--)
    printf("GeeksQuiz ");
  return 0;
}
```

P2 :

```c
int main()
{
    void demo();
    void (*fun)();
    fun = demo;
    (*fun)();
    fun();
    return 0;
}

void demo()
{
    printf("GeeksQuiz ");
}
```

OUTPUT: D(compiler error).
EXPLANATION: p1 gives compiler error(calling before definition) and p2 print GeeksQuizGeeksQuiz.

19) what will be the output of the program?

```c
#include<stdio.h>
int main(){
    int a = 10;
    void *p = &a;
    int *ptr = p;
    printf("%u",*ptr);
    return 0;
}
```

- (A) 10
- (B) Address
- (C) 2
- (D) Compilation error
- (E) None of above

OUTPUT: A(10).
EXPLANATION: we define variable a and general pointer points to this variable the define another integer pointer points tp general pointer, so now integer pointer points to integer variable.

……

20) what will be the output of the program?

```c
int main()
{
 char *ptr = "GeeksQuiz";
 printf("%cn", *&*&*ptr);
 return 0;
}
```

- Compiler Error
- Garbage Value
- Runtime Error
- G
- GeeksQuiz

OUTPUT: G.
EXPLANATION: The operator * is used for dereferencing and the operator & is used to get the address. These operators cancel out effect of each other when used one after another. We can apply them alternatively any no. of times. In the above code, ptr is a pointer to first character of string g. *ptr gives us g, &*ptr gives address of g, *&*ptr again g, &*&*ptr address of g.

21) what will be the output of the program?

```
#include<stdio.h>
void main(){
    static int a=2,b=4,c=8;
    static int *arr1[2]={&a,&b};
    static int *arr2[2]={&b,&c};
    int* (*arr[2])[2]={&arr1,&arr2};
    printf("%d %d\t",*(*arr[0])[1],  *(*(**(arr+1)+1)));
}
```

- (A) 2 4
- (B) 2 8
- (C) 4 2
- (D) 4 8
- (E) None of the above

OUTPUT: D(4 8).
EXPLANATION: No explanation.

……

22) what will be the output of the program?

```
#include<stdio.h>
void fun(int arr[])
{
  int i;
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  for (i = 0; i < arr_size; i++)
      printf("%d ", arr[i]);
}

int main()
{
  int i;
  int arr[4] = {10, 20 ,30, 40};
  fun(arr);
  return 0;
}
```

- 10 20 30 40
- Machine Dependent
- 10 20
- Nothing

OUTPUT: Machine Dependant.
EXPLANATION: ratio of arr/arr[0] not defined as sizeof pointer and integer is compiler dependant.

23) what will be the output of the program?

```c
#include<stdio.h>

int main()
{
    int x=30, *y, *z;
    y=&x; /* Assume address of x is 500 and integer is 4 byte size */
    z=y;
    *y++=*z++;
    x++;
    printf("x=%d, y=%d, z=%d\n", x, y, z);
    return 0;
}
```

- A. x=31, y=502, z=502
- B. x=31, y=500, z=500
- C. x=31, y=498, z=498
- D. x=31, y=504, z=504

OUTPUT:C(31,504,504).
EXPLANATION: we increment pointer by size of integer so we add 4 to the address of x and increment x by one.

……

24) what will be the output of the program?

```c
void swap (char *x, char *y)
{
    char *t = x;
    x = y;
    y = t;
}

int main()
{
    char *x = "geeksquiz";
    char *y = "geeksforgeeks";
    char *t;
    swap(x, y);
    printf("(%s, %s)", x, y);
    t = x;
    x = y;
    y = t;
    printf("n(%s, %s)", x, y);
    return 0;
}
```

OUTPUT: geeksquiz, geeksforgeeks.
geeksforgeeks, geeksquiz.
EXPLANATION: function dosen't swap two strings.
To swap two strings:

```c
void swap1(char **str1_ptr, char **str2_ptr)
{
  char *temp = *str1_ptr;
  *str1_ptr = *str2_ptr;
  *str2_ptr = temp;
}

int main()
{
  char *str1 = "geeks";
  char *str2 = "forgeeks";
  swap1(&str1, &str2);
  printf("str1 is %s, str2 is %s", str1, str2);
  return 0; }
```

25) what will be the output of the program?

```
#include<stdio.h>
int main(){
     int i = 5;
     int *p;
     p = &i;
     printf(" %u %u", *&p , &*p);
     return 0;
}
```

- A) 5 Address
- (B) Address Address
- (C) Address 5
- (D) Compilation error
- (E) None of above

OUTPUT:B(Address,Adress).
EXPLANATION:* and & cancelled each other so we print p which is address.

……

26) what will be the output of the program?

```
# include <stdio.h>
void fun(int x)
{
     x = 30;
}

int main()
{
   int y = 20;
   fun(y);
   printf("%d", y);
   return 0;
}
```

- 30
- 20
- Compiler error
- Runtime error

OUTPUT:20.
EXPLANATION: we pass the variable by value to change the value of y, it must be passed by ref like this->fun(&y).

27)what does the following c statement declare?

```
int ( * f) (int * ) ;
```

- A function that takes an integer pointer as argument and returns an integer.
- A function that takes an integer as argument and returns an integer pointer.
- A pointer to a function that takes an integer pointer as argument and returns an integer.
- A function that takes an integer pointer as argument and returns a function pointer

OUTPUT:C.
Typedef int (*pf)(int)
EXPLANATION:A->int fun(int *), B->int *fun(int), D->(*pf) fun(int *);

……

28) what will be the output of the program?

```
#define print(x) printf("%d ", x)
int x;
void Q(int z)
{
     z += x;
     print(z);
}
void P(int *y)
{
     int x = *y + 2;
     Q(x);
     *y = x - 1;
     print(x);
}
main(void)
{
     x = 5;
     P(&x);
     print(x);
}
```

- 12 7 6
- 22 12 11
- 14 6 6
- 7 6 6

OUTPUT:12 7 6.
EXPLANATION: x is global so first x becomes 5 by the first line in main(). Then main() calls P() with address of x.
// in main(void)

x = 5 // Change global x to 5
P(&x)
P() has a local variable named 'x' that hides global variable. P() theb calls Q() by passing value of local 'x'.
// In P(int *y)

int x = *y + 2; // Local x = 7
Q(x);
In Q(int z), z uses x which is global
// In Q(int z)

z += x; // z becomes 5 + 7
printz(); // prints 12
After end of Q(), control comes back to P(). In P(), *y (y is address of global x) is changed to x - 1 (x is local to P()).
// Back in P()

*y = x - 1; // *y = 7-1
print(x); // Prints 7
After end of Q(), control comes back to main(). In main(), global x is printed.
// Back in main()

print(x); // prints 6 (updated in P()
// by *y = x - 1)

29) what will be the output of the program?

```
int main(){
    int a,b,c,d;
    char *p = ( char *)0;
    int *q = ( int *q)0;
    float *r = ( float *)0;
    double *s = 0;
    a = (int)(p+1);
    b = (int)(q+1);
    c = (int)(r+1);
    d = (int)(s+1);
    printf("%d %d %d %d",a,b,c,d);
return 0;
}
```

- (A) 2 2 2 2
- (B) 1 2 4 8
- (C) 1 2 2 4
- (D) Compilation error
- (E) None of above

OUTPUT:1 2 4 8.
EXPLANATION: we increment p by sizeof char , q with sizeof int , r with sizeof float and s with sizeof double

……

30) what will be the output of the program?

```
#include<stdio.h>
int main(){
   int a = 320;
   char *ptr;
   ptr =( char *)&a;
   printf("%d ",*ptr);
   return 0;
```

- (A) 2
- (B) 320
- (C) 64
- (D) Compilation error
- (E) None of above

OUTPUT:64.
EXPLANATION: Binary of 320 is(10100000), and we define char pointer so pointer will point to one byte.

31)Assume that float takes 4 bytes, predict the output?

```
int main()
{
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;

    printf("%f ", *ptr2);
    printf("%d", ptr2 - ptr1);

   return 0;
}
```

- ○ 90.500000 3
- ○ 90.500000 12
- ○ 10.000000 12
- ○ 0.500000 3

OUTPUT:90.500000 3.
EXPLANATION: ptr1 points to the first element and ptr2 points to the fourth element.

……

32)The following statement in c int (*f())[]; declares?

- ○ a function returning a pointer to an array of integers.
- ○ a function returning an array of pointers to integers.
- ○ array of functions returning pointers to integers.
- ○ an illegal statement.

OUTPUT: int (*f())[ ]; declares a function returning a pointer to an array of integers..
A-> int *(f())[].

33)assume int 4 bytes, char 1 byte and pointer is 4 bytes, what is output?

```
int main()
{
    int arri[] = {1, 2 ,3};
    int *ptri = arri;

    char arrc[] = {1, 2 ,3};
    char *ptrc = arrc;

    printf("sizeof arri[] = %d ", sizeof(arri));
    printf("sizeof ptri = %d ", sizeof(ptri));

    printf("sizeof arrc[] = %d ", sizeof(arrc));
    printf("sizeof ptrc = %d ", sizeof(ptrc));

    return 0;
}
```

- ◯ A) sizeof arri[] = 3 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 4
- ◯ B) sizeof arri[] = 12 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 1
- ◯ C) sizeof arri[] = 3 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 1
- ◯ D) sizeof arri[] = 12 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 4

OUTPUT: D
EXPLANATION: Size of an array is number of elements multiplied by the type of element, that is why we get sizeof arri as 12 and sizeof arrc as 3. Size of a pointer is fixed for a compiler. All pointer types take same number of bytes for a compiler. That is why we get 4 for both ptri and ptrc

……

34) what will be the output of the program?

```
int main(){
    register a = 25;
    int far *p;
    p=&a;
    printf("%d ",*p);
    return 0;
}
```

- ◯ (A) 25
- ◯ (B) 4
- ◯ (C) Address
- ◯ (D) Compilation error
- ◯ (E) None of above

OUTPUT: D(compiler error).
EXPLANATION: We can't use address of register(variable that declare as register store in cpu not memory.)

35) Point out the compile time error in the program given below.

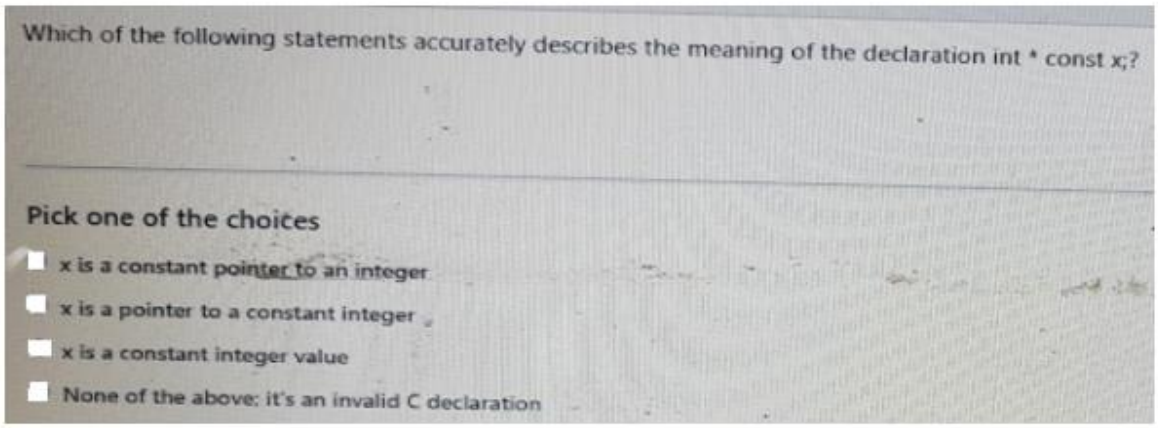
```c
#include<stdio.h>

int main()
{
    int *x;
    *x=100;
    return 0;
}
```

- A. Error: invalid assignment for x
- B. Error: suspicious pointer conversion
- C. No error
- D. None of above

OUTPUT: C(No error).
EXPLANATION: While reading the code there is no error, but upon running the program having an uninitialized variable can cause the program to crash (Null pointer assignment).

.....

36)



- A
- B
- C
- D

OUTPUT: X is constant pointer to an integer.

37)

In the following program add a statement in the function `fact()` such that the factorial gets stored in `j`.

```
#include<stdio.h>
void fact(int*);

int main()
{
    int i=5;
    fact(&i);
    printf("%d\n", i);
    return 0;
}
void fact(int *j)
{
    static int s=1;
    if(*j!=0)
    {
        s = s**j;
        *j = *j-1;
        fact(j);
        /* Add a statement here */
    }
}
```

- A. j=s;
- B. *j=s;
- C. *j=&s;
- D. &j=s;

OUTPUT:B(*j=s)

.....

38)

The following program reports an error on compilation.

```
#include<stdio.h>
int main()
{
    float i=10, *j;
    void *k;
    k=&i;
    j=k;
    printf("%f\n", *j);
    return 0;
}
```

- True
- False

OUTPUT: False

39)

```
#include<stdio.h>
void f(int *p, int *q)
{
   p = q;
   *p = 2;
}
int i = 0, j = 1;
int main()
{
   f(&i, &j);
   printf("%d %d n", i, j);
   getchar();
   return 0;
}
```

- ◯ 2 2
- ◯ 2 1
- ◯ 0 1
- ◯ 0 2

OUTPUT:0 2.
EXPLANATION: /* p points to i and q points to j */
void f(int *p, int *q)
{
 p = q; /* p also points to j now */
 p = 2; / Value of j is changed to 2 now */
}

……

40)

What will be output of following program?

```
#include<stdio.h>
int main(){
    int i = 3;
    int *j;
    int **k;
    j=&i;
    k=&j;
    printf("%u %u %d ",k,*k,**k);
    return 0;
}
```

- ◯ (A) Address, Address, 3
- ◯ (B) Address, 3, 3
- ◯ (C) 3, 3, 3
- ◯ (D) Compilation error
- ◯ (E) None of above

OUTPUT: A(Address,Adress,3).

41) Consider the size of int as two bytes and size of char as one byte. Predict the output of the following code . Assume that the machine is little-endian.

Consider the following C code

```
int main()
{
    int a = 300;
    char *b = (char *)&a;
    *++b = 2;
    printf("%d ",a);
    return 0;
}
```

- ◯ 556
- ◯ 300
- ◯ Runtime Error
- ◯ Compile Time Error

OUTPUT:556.
EXPLANATION: binary for 300(1 0010 1100) and pointer points to first byte and the increment pointer by one so it's point to second byte and assign 2 to second byte to binary will be (10 0010 1100) which is 4+8+32+512=556.

……

42)

Output of following program?

```
#include <stdio.h>

int main()
{
     int *ptr;
     int x;

     ptr = &x;
     *ptr = 0;

     printf(" x = %dn", x);
     printf(" *ptr = %dn", *ptr);

     *ptr += 5;
     printf(" x   = %dn", x);
     printf(" *ptr = %dn", *ptr);

     (*ptr)++;
     printf(" x = %dn", x);
     printf(" *ptr = %dn", *ptr);

     return 0;
}
```

OUTPUT: x = 0, *ptr = 0
x = 5, *ptr = 5
x = 6, *ptr = 6

43)what will be the output of the following code?

```
void main()
{
     struct field
     {
          int a;
          char b;
     }bit;
     struct field bit1={5,'A'};
     char *p=&bit1;
     *p=45;
     clrscr();
     printf("\n%d",bit1.a);
     getch();
}
```

- ◯ 5
- ◯ 45
- ◯ 0
- ◯ None of above

OUTPUT:45.

.....

44) What does the following expression means ? char *(*(* a[N]) ( )) ( );

- ◯ a pointer to a function returning array of n pointers to function returning character pointers.
- ◯ a function return array of N pointers to functions returning pointers to characters
- ◯ an array of n pointers to function returning pointers to characters
- ◯ an array of n pointers to function returning pointers to functions returning pointers to characters.
- ◯ all of them

OUTPUT: all of them.	

.....

45)

Is there any difference between the following two statements?

```
char *p=0;
char *t=NULL;
```

A. Yes

B. No

- ◯ A
- ◯ B

OUTPUT: No.

46)

Pick the best statement for the following program snippet:

```
#include <stdio.h>

int main()
{
 int var;  /*Suppose address of var is 2000 */

 void *ptr = &var;
 *ptr = 5;
 printf("var=%d and *ptr=%d",var,*ptr);

 return 0;
}
```

- It will print "var=5 and *ptr=2000"
- It will print "var=5 and *ptr=5"
- It will print "var=5 and *ptr=XYZ" where XYZ is some random address
- Compile error

OUTPUT: compiler error.
EXPLANATION: we must use casting with general pointer.

……

47)

What will be output of following program?

```
#include<stdio.h>
int main(){
     char arr[10];
     arr = "world";
     printf("%s",arr);
     return 0;
}
```

- (A) world
- (B) w
- (C) Null
- (D) Compilation error
- (E) None of above

OUTPUT:D(Compiler error).
EXPLANATION: arr is constant pointer so we can't make pointer to point another address.to fix this issue use strcpy(arr,"world");

48)

```
int f(int x, int *py, int **ppz)
{
  int y, z;
  **ppz += 1;
   z   = **ppz;
  *py += 2;
   y = *py;
   x += 3;
   return x + y + z;
}

void main()
{
    int c, *b, **a;
    c = 4;
    b = &c;
    a = &b;
    printf("%d ", f(c, b, a));
    return 0;
}
```

- ○ 18
- ○ 19
- ○ 21
- ○ 22

OUTPUT:19
EXPLANATION:
/* below line changes value of c to 5. Note that x remains unaffected by
this change as x is a copy of c and address of x is different from c*/
**ppz += 1
/* z is changed to 5*/
z = **ppz;
/* changes c to 7, x is not changed */
*py += 2;
/* y is changed to 7*/
y = *py;
/* x is incremented by 3 */
x += 3;
/* return 7 + 7 + 5*/

.....

49)

What will be the output produced by the following C code:

```
int main()
{
    int array[5][5];
    printf("%d",( (array == *array) && (*array == array[0]) ));
    return 0;
}
```

OUTPUT:1.

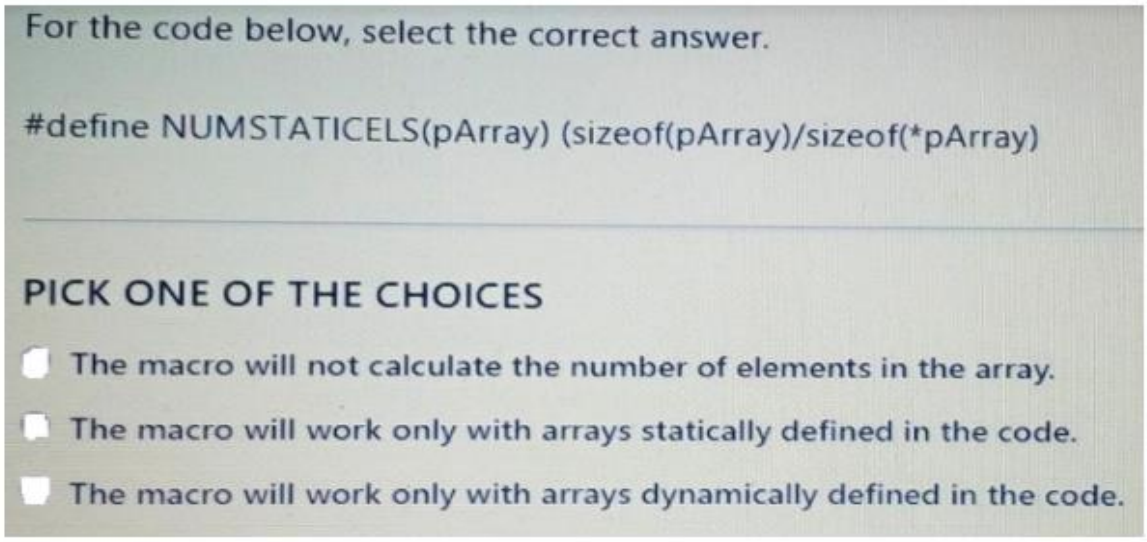
50)

```
void start();
void end();
#pragma startup start
#pragma exit end
int static i;
void main(){
   printf("\nmain function: %d",++i);
}
void start(){
   clrscr();
   printf("\nstart function: %d",++i);
}
void end(){
   printf("\nend function: %d",++i);
   getch();
}
```

OUTPUT: start function:1
Main function:2
End function:3

……

51)



A

B

C

OUTPUT:A.

52)

Q49)Is the NULL pointer same as an uninitialised pointer? *

- ◯ Yes
- ◯ No

OUTPUT:NO.

……

53)

What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    char *str;
    str = "%d\n";
    str++;
    str++;
    printf(str-2, 300);
    return 0;
}
```

- ◯ [A]. No output
- ◯ [B]. 30
- ◯ [C]. 3
- ◯ [D]. 300

OUTPUT:300. EXPLANATION: we increment pointer two times so we print str+2-2=str=”%d”.

54) What would be the equivalent pointer expression for referring the array element a[i][j][k][l].

- A. ((((a+i)+j)+k)+l)
- B. *(*(*(*(a+i)+j)+k)+l)
- C. (((a+i)+j)+k+l)
- D. ((a+i)+j+k+l)

OUTPUT:B.

……

55)

What will be output when you will execute following c code?

```
#include<stdio.h>
typedef struct{
    char *name;
    double salary;
}job;
void main(){
    static job a={"TCS",15000.0};
    static job b={"IBM",25000.0};
    static job c={"Google",35000.0};
    int x=5;
    job * arr[3]={&a,&b,&c};
    printf("%s  %f\t",(3,x>>5-4)[*arr]);
}
double myfun(double d){
        d-=1;
        return d;
}
```

- (A) TCS 15000.000000
- (B) IBM 25000.000000
- (C) Google 35000.000000
- (D) Compilation error
- (E) None of the above

OUTPUT:C.
EXPLANATION: (3,x>>1)->(3,2)[*arr], comma here is an operator so we get 2[*arr] =*arr[2],so structure c will print.

56)

What will be the output of the program?

```
#include<stdio.h>

int main()
{
    int arr[3] = {2, 3, 4};
    char *p;
    p = arr;
    p = (char*)((int*)(p));
    printf("%d, ", *p);
    p = (int*)(p+1);
    printf("%d", *p);
    return 0;
}
```

- [A]. 2, 3
- [B]. 2, 0
- [C]. 2, Garbage value
- [D]. 0, 0

OUTPUT:B.
EXPLANATION: pointer casting to char so first printf will print first byte which is 2 and then pointer increment by one so will point to second byte which is zero.

……

57)

What will be output when you will execute following c code?

```
#include<stdio.h>
void main(){
    short num[3][2]={3,6,9,12,15,18};
    printf("%d  %d",*(num+1)[1],**(num+2));
}
```

- (A) 12 18
- (B) 18 18
- (C) 15 15
- (D) 12 15
- (E) Compilation error

OUTPUT:15 15.
EXPLANATION: we have two dimensional array so num will points to the first row(3,6) and num+1 will point to second row(9,12) and num+2 will point to third row(15,18).

58)

Consider this C code to swap two integers and these five statements after it:

```
void swap(int *px, int *py)
{
    *px = *px - *py;
    *py = *px + *py;
    *px = *py - *px;
}
```

S1: will generate a compilation error S2: may generate a segmentation fault at runtime depending on the arguments passed S3: correctly implements the swap procedure for all input pointers referring to integers stored in memory locations accessible to the process S4: implements the swap procedure correctly for some but not all valid input pointers S5: may add or subtract integers and pointers.

- ◯ S1
- ◯ S2 and S3
- ◯ S2 and S4
- ◯ S2 and S5

OUTPUT:s2 and s4.
EXPLANATION: S2: May generate segmentation fault if value at pointers px or py is constant or px or py points to a memory location that is invalid S4: May not work for all inputs as arithmetic overflow can occur

……

59)

Which of the statements is correct about the program?

```
#include<stdio.h>

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

- ◯ [A]. Output: Garbage value
- ◯ [B]. Output: 1
- ◯ [C]. Output: 3
- ◯ [D]. Error: Invalid indirection

OUTPUT:D.

60)

What will be the output of the program ?

```
#include<stdio.h>
int *check(static int, static int);

int main()
{
    int *c;
    c = check(10, 20);
    printf("%d\n", c);
    return 0;
}
int *check(static int i, static int j)
{
    int *p, *q;
    p = &i;
    q = &j;
    if(i >= 45)
        return (p);
    else
        return (q);
}
```

- A. 10
- B. 20
- C. Error: Non portable pointer conversion
- D. Error: cannot use static for function parameters

OUTPUT:D.(we can't use any type of storage classes as parameter except register).

……

61)

What will be output of following program?

```
#include<stdio.h>
int main(){
     int * p , b;
     b = sizeof(p);
     printf("%d" , b);
return 0;
}
```

- (A) 2
- (B) 4
- (C) 8
- (D) Compilation error
- (E) None of above

OUTPUT:4

62)

```c
#include "stdio.h"
int main()
{
 void *pVoid;
 pVoid = (void*)0;
 printf("%lu",sizeof(pVoid));
 return 0;
}
```

- Assigning (void *)0 to pVoid isn't correct because memory hasn't been allocated. That's why no compile error but it'll result in run time error.
- Assigning (void *)0 to pVoid isn't correct because a hard coded value (here zero i.e. 0) can't assigned to any pointer. That's why it'll result in compile error.
- No compile issue and no run time issue. And the size of the void pointer i.e. pVoid would equal to size of int.
- sizeof() operator isn't defined for a pointer of void type.

OUTPUT:C.	

.....

63)

What will be output of following program?

```c
#include<stdio.h>
#include<string.h>
int main(){
     char *ptr1 = NULL;
     char *ptr2 = 0;
     strcpy(ptr1," c");
     strcpy(ptr2,"questions");
     printf("\n%s %s",ptr1,ptr2);
     return 0;
}
```

- (A) c questions
- (B) c (null)
- (C) (null) (null)
- (D) Compilation error
- (E) None of above

OUTPUT:E--→segmentation fault EXPLANATION: to solve this issue we use ptr1="C" and ptr2="question".

64)

Will the program compile in Turbo C?

```
#include<stdio.h>
int main()
{
    int a=10, *j;
    void *k;
    j=k=&a;
    j++;
    k++;
    printf("%u %u\n", j, k);
    return 0;
}
```

- ◯ Yes
- ◯ No

OUTPUT: No.

.....

65)

What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    static char *s[] = {"black", "white", "pink", "violet"};
    char **ptr[] = {s+3, s+2, s+1, s}, ***p;
    p = ptr;
    ++p;
    printf("%s", **p+1);
    return 0;
}
```

- ◯ [A] ink
- ◯ [B]. ack
- ◯ [C]. ite
- ◯ [D]. let

OUTPUT: ink.
EXPLANATION: p points to “violet” and then increment pointer so it points to pink and we skip the first letter then print the rest of pink which is ink

66)

What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    int i, a[] = {2, 4, 6, 8, 10};
    change(a, 5);
    for(i=0; i<=4; i++)
        printf("%d, ", a[i]);
    return 0;
}
void change(int *b, int n)
{
    int i;
    for(i=0; i<n; i++)
        *(b+1) = *(b+i)+5;
}
```

- ◯ [A]. 7, 9, 11, 13, 15
- ◯ [B]. 2, 15, 6, 8, 10
- ◯ [C]. 2 4 6 8 10
- ◯ [D]. 3, 1, -1, -3, -5

OUTPUT: C.
EXPLANATION: we change in second element only.

……

67)

Q18)What's the size returned for each of sizeof() operator? * 1

Assume *int* is 4 bytes, *char* is 1 byte and *float* is 4 bytes. Also, assume that pointer size is 4 bytes (i.e. typical case)

```
char *pChar;
int *pInt;
float *pFloat;

sizeof(pChar);
sizeof(pInt);
sizeof(pFloat);
```

- ◯ 4 4 4
- ◯ 1 4 4
- ◯ 1 4 8
- ◯ None of the above

OUTPUT:4 4 4 .
EXPLANATION: Sizeof pointer(any type) is 4 byte for 32bit architecture and 8 byte for 64bit architecture

68)

```c
#include<stdio.h>
int main()
{
    int a = 12;
    void *ptr = (int *)&a;
    printf("%d", *ptr);
    getchar();
    return 0;
}
```

OUTPUT: compiler error.
EXPLANATION: we can't dereference void pointer without casting.

……

69)

What will be output of following program?

```c
#include<stdio.h>
unsigned long int (* avg())[3]{
    static unsigned long int arr[3] = {1,2,3};
    return &arr;
}
int main(){
    unsigned long int (*ptr)[3];
    ptr = avg();
    printf("%d" , *(*ptr+2));
    return 0;
}
```

OUTPUT: 3.
EXPLANATION: we print the last element in array.

ADDITIONAL TRICKS

1) What will be the output of the program ?

```c
#include<stdio.h>

int main()
{
    char str[20] = "Hello";
    char *const p=str;
    *p='M';
    printf("%s\n", str);
    return 0;
}
```

A. Mello

B. Hello

C. HMello

D. MHello

OUTPUT: Mello.
EXPLANATION: We change the first byte in string

.....

2) What will be the output of the program ?

```c
#include<stdio.h>

int main()
{
    char *str;
    str = "%s";
    printf(str, "K\n");
    return 0;
}
```

A. Error

B. No output

C. K

D. %s

OUTPUT: K.
EXPLANATION: we replace str in printf function with

3) What will be the output of the program ?

```c
#include<stdio.h>

int main()
{
    printf("%d, %d\n", sizeof(NULL), sizeof(""));
    return 0;
}
```

A. 2, 1

B. 2, 2

C. 4, 1

D. 4, 2

OUTPUT: 4 1.
EXPLANATION: In Turbo C, the output will be 2, 1 because the size of the pointer is 2 bytes in 16-bit platform.
But in Linux, the output will be 4, 1 because the size of the pointer is 4 bytes.
This difference is due to the platform dependency

.....

4) What will be the output of the program ? 74 is ascii of J and 65 is ascii of A.

```c
#include<stdio.h>

int main()
{
    void *vp;
    char ch=74, *cp="JACK";
    int j=65;
    vp=&ch;
    printf("%c", *(char*)vp);
    vp=&j;
    printf("%c", *(int*)vp);
    vp=cp;
    printf("%s", (char*)vp+2);
    return 0;
}
```

A. JCK

B. J65K

C. JAK

D. JACK

OUTPUT: JACK.
EXPLANATION: first we assign address of ch to void pointer then print the value of ch(74 is ascii of J) and so on.

5) What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    char str1[] = "India";
    char str2[] = "BIX";
    char *s1 = str1, *s2=str2;
    while(*s1++ = *s2++)
        printf("%s", str1);

    printf("\n");
    return 0;
}
```

A. IndiaBIX

B. BndiaBIdiaBIXia

C. India

D. (null)

OUTPUT: B.
EXPLANATION: it's easy :D.

.....

6) What will be the output of the program ?

```
#include "stdio.h"
int main()
{
    char a[] = { 'A', 'B', 'C', 'D' };
    char* ppp = &a[0];
    *ppp++; // Line 1
    printf("%c %c ", *++ppp, --*ppp); // Line 2
}
```

OPTIONS:

a)C B

b)B A

c)B C

d)C A

OUTPUT: C A.
EXPLANATION: Line 1 : Now, ppp points to next memory location i.e., index 1 of the character array.
Line 2 : Firstly, –*ppp= –(*ppp) is executed and hence the value 'B' (which is in the index 1 position of the char[] array) gets decremented by 1(i.e., it becomes 'A')and it is sent for printing. Then *++ppp= *(++ppp) is executed which initially increments the pointer to the next element of the array and prints the value in that index number 2 which is 'C'. Although –*ppp is executed first compared to *++ppp, the display will be shown in the order as we mentioned in the printf() function in line 2. Hence we get output as C A.

7) What will be the output of the program ?

```
int main()
{
    int x[5] = { 1, 2, 3, 4, 5 };
    // p points to array x
    int* p = x;
    int i;
    // exchange values using pointer
    for (i = 0; i < 2; i++) {
        int temp = *(p + i);
        *(p + i) = *(p + 4 - i);
        *(p + 4 - i) = temp;
    }
    // output the array x
    for (i = 0; i < 5; i++)
        cout << x[i] << " ";
    return 0;
}
```

a) 5 4 3 2 1

b) 1 2 3 4 5

c)Address of the elements

d) Can't say

OUTPUT: 5 4 3 2 1.
EXPLANATION: p points to array x then exchange values using pointer in the for loop to after exchanging value the last loop print the value.

.....

8)

The reason for using pointers in a Cprogram is

A. Pointers allow different functions to share and modify their local variables.

B. To pass large structures so that complete copy of the structure can be avoided.

C. Pointers enable complex "linked" data structures like linked lists and binary trees.

D. All of the above

OUTPUT: All of the above.

9) What will be the output of the program ?

```c
int main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int *p = arr;
    ++*p;
    p += 2;
    printf("%d", *p);
    return 0;
}
```

A 2

B 3

C 4

D Compiler Error

OUTPUT:3.
EXPLANATION: The expression ++*p is evaluated as "++(*p)" . So it increments the value of first element of array (doesn't change the pointer p). When p += 2 is done, p is changed to point to third element of array.

.....

10) What will be the output of the program ?

```c
void f(char**);
int main()
{
    char *argv[] = { "ab", "cd", "ef", "gh", "ij", "kl" };
    f(argv);
    return 0;
}
void f(char **p)
{
    char *t;
    t = (p += sizeof(int))[-1];
    printf("%sn", t);
}
```

A ab

B cd

C ef

D gh

OUTPUT: gh
EXPLANATION: p+=sizeof(int)->p=p+4[-1]->p=p+4-3->p=p+3 which is the third element in array.

11) What will be the output of the program ?

```
void mystery(int *ptra, int *ptrb)
{
    int *temp;
    temp = ptrb;
    ptrb = ptra;
    ptra = temp;
}
int main()
{
     int a=2016, b=0, c=4, d=42;
     mystery(&a, &b);
     if (a < c)
        mystery(&c, &a);
     mystery(&a, &d);
     printf("%dn", a);
}
```

A)2016.

B)0.

C)4.

D)8.

OUTPUT: 2016.
EXPLANATION: Note that a and d are not swapped as the function mystery() doesn't change values, but pointers which are local to the function.

……

12) What will be the output of the program ?

```
void printxy(int x, int y)
{
    int *ptr;
    x = 0;
    ptr = &x;
    y = *ptr;
    *ptr = 1;
    printf("%d,%d", x, y);
}
```

The output of the printxy(1,1) is

A 0,0

B 0,1

C 1,0

D 1,1

OUTPUT: 1,0.
EXPLANATION: we define integer pointer and reassign x with 0 then assign address of x to ptr then assign the value of x(*ptr) to y so y now is contain 0 the change the value that pointer points to(x) to 1 so x has value 1.

13)

'ptrdata' is a pointer to a data type. The expression *ptrdata++ is evaluated as (in C++) :

A *(ptrdata++)

B (*ptrdata)++

C *(ptrdata)++

D Depends on compiler

OUTPUT: A.
EXPLANATION: Here ++ (unary operator) have high precedence than * (unary operator). So ++ will be evaluated first then * will be in action. So A is the correct option. So, option (A) is correct.

……

14) What will be the output of the program ?

```
main()
{
char g[] = "geeksforgeeks";
printf("%s", g + g[6] - g[8]);
}
```

A geeks

B rgeeks

C geeksforgeeks

D forgeeks

OUTPUT: A.
EXPLANATION: g[6]-g[8]=8->g+8, so pointer points to the eight's element in array.

15) What will be the output of the program ?

```c
#include <stdio.h>

int main()
{

    char *pchar="aticle";

    pchar[1]='d';

    printf("%c",pchar[1]);

    return 0;
}
```

A)d.

B)a.

c)syntax error.

d)runtime error.

OUTPUT: d.
EXPLANATION: The “aticle” is a string literal, so when to try to compile the above program compiler does not throw the error but when you try to run the program. it will be the crash.

……

16) What will be the output of the program ?

```c
#include <stdio.h>

int main()
{
    const char *pcName="aticleworld";

    pcName[0] = 'A' ;
    printf("%s", pcName);

    return 0;
}
```

OUTPUT: [Error] assignment of read-only location ‘*pcName’

EXPLANATION: Here pointer points to character constant so we cannot replace the character.

17) What will be the output of the program ?

```c
#include <stdio.h>

int main()
{
    char * const pcName="aticleworld";

    pcName++;
    printf("%s",pcName);

    return 0;
}
```

OUTPUT: [Error] increment of read-only variable ‘pcName’.

EXPLANATION: In the above example pointer itself constant so we cannot increment the pointer.

……

18) What will be the output of the program ?

```c
#include <stdio.h>

int main()
{
    const int ciData = 5;
    int * piData = NULL;

    printf("Value before the change = %d\n\n",ciData);

    //assign the constant address to the pointer
    piData = (int*)&ciData;

    *piData = 6;
    printf("Value after the change = %d\n\n",ciData);

    return 0;
}
```

OUTPUT: Value before the change =5
Value after the change = 6

EXPLANATION: ciData stores in stack so we can change it’s value indirect by using pointers.

19) What will be the output of the program ?

```c
#include <stdio.h>

int main()
{
    const int a =7;
    const int  * p=&a;

    printf("%d",*++p);

    return 0;
}
```

OUTPUT: GarbageData.
EXPLANATION: in the above example, the indirection operator and pre-increment operator have the same priority level and associative is left-right. So *++p behave like *(++p) that is why now p point to a garbage data location.

……

20) What will be the output of the program ?

```c
#include <stdio.h>

int main()
{
    const int a =7;
    const int  * p=&a;

    printf("%d",++*p);

    return 0;
}
```

OUTPUT: [Error] increment of read-only location '*p'.
EXPLANATION: In the above example, the indirection operator and pre-increment operator have the same priority level and associative is left-right. Here ++*p behave like ++(*p), so when to compile this code compiler throws a compilation error because of pointer p point to a read-only address.

……

21)how to pass and return one dimensional array?

Pass Array:

```c
#include <stdio.h>
float calculateSum(float age[]);

int main() {
    float result, age[] = {23.4, 55, 22.6, 3, 40.5, 18};

    // age array is passed to calculateSum()
    result = calculateSum(age);
    printf("Result = %.2f", result);
    return 0;
}

float calculateSum(float age[]) {

  float sum = 0.0;

  for (int i = 0; i < 6; ++i) {
                sum += age[i];
  }

  return sum;
}
```

Return Array:

```c
#include <stdio.h>
/* function to generate and return random numbers */
int * getRandom( ) {

   static int  r[10];
   int i;

   /* set the seed */
   srand( (unsigned)time( NULL ) );

   for ( i = 0; i < 10; ++i) {
      r[i] = rand();
      printf( "r[%d] = %d\n", i, r[i]);
   }

   return r;
}
/* main function to call above defined function */
int main () {

   /* a pointer to an int */
   int *p;
   int i;

   p = getRandom();

   for ( i = 0; i < 10; i++ ) {
      printf( "*(p + %d) : %d\n", i, *(p + i));
   }

   return 0;
}
```

22) how to pass and return two dimensional array?

Pass Array:

a)When both dimensions are available globally (either as a macro or as a global constant).

```
#include <stdio.h>
#define M 3
#define N 3
void print(int arr[M][N])
{
    int i, j;
    for (i = 0; i < M; i++)
      for (j = 0; j < N; j++)
        printf("%d ", arr[i][j]);
}

int main()
{
    int arr[][M] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    print(arr);
    return 0;
}
```

b)Using pointer to whole array:

```
#include <stdio.h>
#define M 3
#define N 3
void print(int (*arr)[M][N])
{
    int i, j;
    for (i = 0; i < M; i++)
      for (j = 0; j < N; j++)
        printf("%d ", (*arr)[i][j]);
}

int main()
{
    int arr[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    print(&arr);
    return 0;
}
```

*arr++ means the pointer will move to a place you don't know because pointer will increase by size of hole array because you sent address of whole array

c)Using pointer to one dimensional array:

```
#include <stdio.h>
#define M 3
#define N 3
void print(int (*arr)[N])
{
    int i;
    for (i = 0; i < M*N; i++)
        printf("%d ",(*arr)[i]);
}

int main()
{
    int arr[][M] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    print(arr);
    return 0;
}
```

*arr++ means the pointer will move to a place of second row "4" because pointer will increase by size of row of array

d)using pointer to integer:

```
#include <stdio.h>
#define M 3
#define N 3
void print(int *arr)
{
    int i;
    for (i = 0; i < M*N; i++)
        printf("%d ",arr[i]);
}

int main()
{
    int arr[][M] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    print(&arr[0][0]);
    return 0;
}
```

```
for(i=0; i<M*N;i++)
{
printf("%d ",*arr);
arr++;}
}
```

……

Return two dimensional array from function:

1)receive in pointer to integer:

```
#include <stdio.h>
#define M 2
#define N 2
int *myfun()
{
   static int a[M][N]={{1,2}, {3,4} };
   return &a[0][0];
}
int main()
{
   int *p=myfun();
   int i,j;
    for(i=0; i<M*N; i++)
    printf("%d ",p[i]);
}
```

2)Reciceve in structure:

```
#include <stdio.h>
#define M 3
#define N 3
struct mystruct {
    int a[M][N];
};

struct mystruct retArr() {
    struct mystruct Smystruct = {
        {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        }
    };

    return Smystruct;
}

int main(int argc, const char* argv[]) {
    struct mystruct Smystruct = retArr();
    int i,j;
    for(i=0; i<M; i++)
          for(j=0; j<N; j++)
          printf("%d ",Smystruct.a[i][j]);
    return 0;
}
```

3)Using pointer to structure:

```
#include <stdio.h>
#define M 3
#define N 3
struct mystruct {
    int a[M][N];
};

struct mystruct retArr() {
    struct mystruct Smystruct = {
        {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        }
    };

    return Smystruct;
}

int main(int argc, const char* argv[]) {
    struct mystruct Smystruct=retArr();
     struct mystruct *pSmystruct=&Smystruct ;
    int i,j;
    for(i=0; i<M; i++)
          for(j=0; j<N; j++)
          printf("%d ",pSmystruct->a[i][j]);
    return 0;
}
```

.....

Using array of pointer to structure:

```
#include <stdio.h>
#define NO_OF_ELEMENTS 2
struct mystruct {
    int age;
    char name[20];
};
int main()
{
    int i;
     struct mystruct Smystruct[NO_OF_ELEMENTS]={{25,"mohamed"},{40,"gamal"}};
     struct mystruct *pSmystruct[NO_OF_ELEMENTS];
     pSmystruct[i] = &Smystruct[i];
     for ( i = 0; i < NO_OF_ELEMENTS; i++)
     {
         pSmystruct[i]=&Smystruct[i];
         pSmystruct[i]=&Smystruct[i];
   }
     for ( i = 0; i < NO_OF_ELEMENTS; i++)
     {
      printf("%d %s\n",pSmystruct[i]->age,pSmystruct[i]->name );
    }
    return 0;
}
```

Using array of pointer to function instead of switch case:

Using switch case:

int (*ptrf[])(int , int)
array of pointers to function is used instead of switch cases

```c
#include <stdio.h>
int addnum(int a, int b)
{
    return a+b;
}
int subnum(int a, int b)
{
    return a-b;
}
int mulnum(int a, int b)
{
    return a*b;
}
int main()
{
 int choice,num1,num2,ret;
 printf("enter choice 0 for add and 1 for sub and 2 for mul:");
 scanf("%d",&choice);
 printf("enter data1:");
 scanf("%d",&num1);
 printf("enter data2:");
 scanf("%d",&num2);
 switch(choice)
 {
    case 0:
      ret=addnum(num1,num2);
      break;
    case 1:
      ret=subnum(num1,num2);
      break;
     case 2:
      ret=mulnum(num1,num2);
      break;
 }
 printf("%d",ret);
}
```

Using array of pointer to function:

```c
int addnum(int a, int b)
{
    return a+b;
}
int subnum(int a, int b)
{
    return a-b;
}
int mulnum(int a, int b)
{
    return a*b;
}
int main()
{
 int choice,num1,num2;
 int (*pf[3])(int,int)={addnum,subnum,mulnum};
 printf("enter choice 0 for int and 1 for sub and 2 for mul:");
 scanf("%d",&choice);
 printf("enter data1:");
 scanf("%d",&num1);
 printf("enter data2:");
 scanf("%d",&num2);
 int ret=(*pf[choice])(num1,num2);
 printf("%d",ret);
    return 0;
}
```

Pointer to function as argument:

Sorting array using pointer to function as argument:

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

}
int compare(int a, int b)
{
     if(a>b) return 1;
     return -1;
}
void sort_array(int *a, int size, int(*compare)(int,int))
{
    int i,j;
    for(i=0; i<size-1; i++)
    {
        for(j=i+1; j<size; j++)
        {
            if(compare(a[i],a[j])>0)
            {
                swap(&a[i],&a[j]);
            }
        }
    }
}
int main()
{
   int a[]={5,4,7,1,9,15,0},size,i;
   size=sizeof(a)/sizeof(*a);
   sort_array(a,size,compare);
   for(i=0; i<size; i++)
    printf("%d ",a[i]);
    return 0;
}
```