2 What is void pointer ? show its use.

Void Pointers

- Generally a pointer variable is declared using data type. Where Data type indicates type of variable whose address can be stored in the pointer variable.
- **For Example :**

```
int a;
float b;
int *ptr;
ptr=&a; // Valid
ptr=&b; // Not Valid
```

Here, pointer is declared as integer. So it can be assigned an address of integer variable. If you are trying to assign an address of float variable then it will generate compile error.

- Thus a pointer variable declared using a particular data type can not hold the address of variables of other data types. This problem can be solved using concept of **Void Pointer.**
- **Void Pointer** is a general purpose pointer which can be used to store address of any variable of any data type (char, int, float etc.) to a void pointer variable.
- General syntax for declaring void pointer is given below:

```
void *PointerName;
```

- **Example :**

```
int a=5;
float b=5.5;
void *ptr;
ptr = &a; // Valid
ptr=&b; // Valid
```

❖ **Dereferencing Void Pointer**

❖ The indirection operator * is known as dereferencing operator.

❖ It is used to retrieve the value of the variable whose address is stored in the pointer.

❖ In case of void pointer we need to type cast the pointer variable to dereference it. Thus in order to retrieve the value of the variable whose address is stored in void pointer we have to perform type casting.

❖ **Example :**

```
#include<stdio.h>
#include<conio.h>
void  main()
{
    int  a  =  5;
    float  b  =  5.5;
    void  *ptr;
    ptr = &a; //Assign address of integer variable to void pointer.
    printf ("Value of Integer Variable is %d", *( (int *) ptr));
    ptr = &b; // Assign address of float variable to void pointer
    printf ("Value of Float variable is %f", *( (float *) ptr));
    getch  ();
}
```

3. What is array of pointer ? explain with example.

Array of Pointers

❖ Array is a collection of variables of same data type. Thus array is used to store more then one values of same data types.

❖ Similarly array of pointer is a collection of addresses. This contains addresses of more then one variables of same data types inside it.

❖ **Example :**

```
#include <stdio.h>
#include <conio.h>
void  main()
{
    clrscr  ();
    int *array[3];
    int x = 15, y = 25, z = 35;
    int i;
```

```
    array[0] = &x;
    array[1] = &y;
    array[2] = &z;
    for (i=0; i< 3; i++)
    {
    printf("The value of %d= %d and address is %u\n", i,
    *(array[i]), array[i]);
    }
    getch();
}
```

❖ **Output :**

The value of 0 = 15 and address is 55248

The value of 1 = 25 and address is 55250

The value of 2 = 35 and address is 55252

(4) List various category of user define functions. Explain any one.

Category of function

❖ Following are the various categories of function:

(1) Function with no return value and no argument list.

o The general syntax of function with no return value and no argument list is given below:

```
void Function_Name ();
```

- **Example :**

```
#include <stdio.h>
#include <conio.h>
void area ();
void main ()
{
    clrscr ();
    area ();
    getch ();
}
void area ()
{
    float a, r;
    printf ("Enter Radius");
    scanf ("%f", &r);
    a = 3.14 * r * r;
    printf ("Area of Circle is:%f",a);
}
```

- **Output :**

Enter Radius:
3
Area of circle is 28.26

- In above example function area () does not accept any arguments and does not return any value to the calling function.

(2) Function with one return value and no argument list.

- The general syntax of function with one return value and no argument list is given below:

Return Type Function_Name ();

- **Example :**

```
#include <stdio.h>
#include <conio.h>
float area ();
void main ()
{
    clrscr ();
    printf ("Area of circle is %f", area ());
    getch ();
}
```

```
float  area  ()
{
    float  a,  r;
    printf  ("Enter  Radius");
    scanf  ("%f",  &r);
    a  =  3.14  *  r  *  r;
    return  a;
}
```

- **Output :**

```
Enter Radius:
3
Area of circle is 28.26
```

- In above example function area () does not accept any arguments but it returns a value of type float to the calling function.

(3) Function with no return value and with argument list.

- The general syntax of function with no return value but with argument list is given below:

```
void Function_Name (Argument List);
```

- **Example :**

```
#include  <stdio.h>
#include  <conio.h>
void  area  (float  r);
void  main  ()
{
    float  r;
    clrscr  ();
    printf  ("Enter  Radius:");
    scanf  ("%f",  &r);
    area  (r);
    getch  ();
}
void  area  (float  x)
{
    float  a;
    a  =  3.14  *  x  *  x;
    printf  ("Area  of  circle  is  %f",  a);
}
```

- **Output :**

```
Enter Radius:
3
```

Area of circle is 28.26

- In above example function area () accepts one argument of type float and does not return any value to the calling function.

(4) Function with one return value and argument list.

- The general syntax of function with one return value and with argument list is given below:

  Return Type Function_Name_(Argument List);

- **Example :**

```
#include <stdio.h>
#include <conio.h>
float area (float r);
void main ()
{
    float r, a;
    clrscr ();
    printf ("Enter Radius:");
    scanf ("%f", &r);
    a = area (r);
    printf ("Area of circle is %f", a);
    getch ();
}
float area (float x)
{
    return (3.14 * x * x);
}
```

- **Output :**

Enter Radius:
3
Area of circle is 28.26

- In above example function area () accepts one argument of type float and returns a value of type float to the calling function.