PRACTICAL 2

1) Write a C Program to perform addition on 2 complex number using structure. (Use user defined function addition).

```
#include <stdio.h>
typedef struct complex {
   float real;
   float imag;
} complex;
complex addition(complex n1, complex n2);
int main() {
   complex n1, n2, result;
   printf("Enter the real and imaginary parts of two numbers: \n");
   scanf("%f %f", &n1.real, &n1.imag);
   scanf("%f %f", &n2.real, &n2.imag);
   result = addition(n1, n2);
   printf("---------------------------------\n");
   printf("Sum = %.1f + %.1fi", result.real, result.imag);  }
complex addition(complex n1, complex n2) {
   complex temp;
   temp.real = n1.real + n2.real;
   temp.imag = n1.imag + n2.imag;
   return (temp);
}
```

OUTPUT

```
Enter the real and imaginary parts of two numbers:
5 2
7 3
---------------------------------
Sum = 12.0 + 5.0i
```

2) Write a C Program to implement searching on unordered array of 10 integer value.

```
#include<stdio.h>
int main() {
   int arr[10], x, count=0;
   printf("Enter any 10 numbers: \n");
   for(int i=0; i<10; i++) {
      scanf("%d",&arr[i]);
   }
   printf("Enter number that you want to search: \n");
   scanf("%d", &x);
   for(int i=0; i<10; i++)  {
      if(arr[i]==x)  {
         printf("-------------------------\n");
         printf("The number is at index: %d\n", i);
      }
   }
}
```

OUTPUT

```
Enter any 10 numbers:
2 4 6 8 1 3 5 7 9 10
Enter number that you want to search:
3
-------------------------
The number is at index: 5
```

3) Write a C Program to find largest value from array of 10 integers.

```
#include<stdio.h>
int main()  {
   int arr[10];
   printf("Enter any 10 numbers: \n");
   for(int i=0; i<10; i++) {
      scanf("%d",&arr[i]);
   }
   for(int i=0; i<10; i++) {
      printf("%d\t",arr[i]);
   }
   int max=arr[0];
   for(int i=0; i<10; i++)  {
      if(arr[i]>max){
         max=arr[i];   }
   }
   printf("\n-----------------------------------------------------------------\n");
   printf("The Largest number from this array is : %d\n", max);
   printf("\n-----------------------------------------------------------------\n");
   return 0;
}
```

OUTPUT

```
Enter any 10 numbers:
1 2 3 4 5 6 7 8 9 10
1       2       3       4       5       6       7       8       9       10
-----------------------------------------------------------------
The Largest number from this array is : 10
-----------------------------------------------------------------
```

4) Write a C Program to Swap Max and Min Value from array of 10 integer value.

```
#include<stdio.h>

int main()

{

   int arr[10], max=0, min=arr[0],temp;

   printf("Enter any 10 integer:  \n");

   for(int i=0; i<10; i++) //input

   {

      scanf("%d",&arr[i]);

   }


   for(int i=0; i<10; i++) //check max value

   {

      if(arr[i]>max)

      {

         max=arr[i];

      }

   }

   for(int i=0; i<10; i++) //check min value

   {

      if(arr[i]<min)

      {

         min=arr[i];

      }

   }

   printf("\n------------------------------------------------------\n");

   printf("Before swapping maximum = %d\tminimum = %d\n", max, min);

   temp=max; //swapping
```

```
    max=min;
    min=temp;
    printf("\n----------------------------------------------------\n");
    printf("After swapping maximum = %d\tminimum = %d\n", max, min);
}
```

OUTPUT

```
Enter any 10 integer:
10 20 30 40 50 60 70 80 90 100
----------------------------------------------------
Before swapping maximum = 100   minimum = 10
----------------------------------------------------
After swapping maximum = 10     minimum = 100
```

5) Write a C Program to insert a value in array of 10 integers at specific position.

```
#include<stdio.h>

int main()

{

   int arr[11], value, pos;

   printf("Enter any 10 numbers: \n"); //input 10 number

   for(int i=0; i<10; i++)

   {

      scanf("%d", &arr[i]);

   }

   printf("----------------------------------------------\n");

   printf("Enter position:\t"); //position at which you want to add number

   scanf("%d", &pos);

   printf("Enter value:\t"); //number that you want to add

   scanf("%d", &value);

   for(int i=10; i>=pos; i--) //shifting to next side

   {

      arr[i]=arr[i-1];

   }

   arr[pos-1]=value;

   printf("----------------------------------------------\n");

   printf("----------------------------------------------\n");

   for(int i=0; i<11; i++) //output  {

      printf("%d\t",arr[i]);

   }

}
```

OUTPUT

```
Enter any 10 numbers:
1 2 3 4 5 6 7 8 9 10
---------------------------------------------
Enter position: 3
Enter value:    90
---------------------------------------------
---------------------------------------------
1     2     90    3     4     5     6     7     8     9     10
```

6) Write a C Program to delete a value in array of 10 integers from specific position.

```
#include<stdio.h>

int main()

{

   int arr[10], pos;

   printf("Enter any 10 numbers: \n"); //input

   for(int i=0; i<10; i++){

      scanf("%d", &arr[i]);

   }

   printf("Enter position: "); //position at which you want to delete element

   scanf("%d", &pos);

   int temp=arr[pos-1]; //here we make pos empty

   for(int i=pos; i<=9; i++) //shifting  {

      arr[i-1]=arr[i];

   }

   printf("\n----------------------------------------------------------------\n");

   for(int i=0; i<9; i++){ //output

      printf("%d\t", arr[i]);

   }

}
```

OUTPUT

```
Enter any 10 numbers:

1 2 3 4 5 6 7 8 9 10

Enter position: 4


----------------------------------------------------------------

1       2       3       5       6       7       8       9       10
```

Question 1: What do you mean by nested structure?

A nested structure in C is a structure within structure. One structure can be declared inside another structure in the same way structure members are declared inside a structure.

Syntax:

```
struct name_1
{
   member1;
  member2;
  .
  .
  membern;
  struct name_2
  {
      member_1;
      member_2;
      .
      .
      member_n;
  }, var1
} var2;
```

Question 2: What is embedded C programming and how it is different from C programming?

Embedded C is generally used to develop microcontroller-based applications. C is a high-level programming language. Embedded C is just the extension variant of the C language. On the other hand, embedded C language is truly hardware dependent. The compilers in embedded C are OS independent. Here, we need a specific compiler that can help in generating micro-controller based results. Famous compilers used in embedded C are BiPOM Electronic, Green Hill Software and more.

Question 3: Explain reference and de-reference operator in pointers.

The reference operator (@expression) lets you refer to functions and variables indirectly, by name. It uses the value of its operand to refer to variable, the fields in a record, function, method, property or child window.

A dereference operator, which is also known as an indirection operator, operates on a pointer variable. It returns the location value, or l-value in memory pointed to by the variable's value. The deference operator is denoted with an asterisk (*).

Question 4: Explain 3 types of pointers.

Null Pointer

You create a null pointer by assigning the null value at the time of pointer declaration. This method is useful when you do not assign any address to the pointer. A null pointer always contains value 0.

Void Pointer

It is a pointer that has no associated data type with it. A void pointer can hold addresses of any type and can be typecast to any type. It is also called a generic pointer and does not have any standard data type. It is created by using the keyword void.

Wild Pointer

Wild pointers are also called uninitialized pointers. Because they point to some arbitrary memory location and may cause a program to crash or behave badly. This type of C pointer is not efficient. Because they may point to some unknown memory location which may cause problems in our program. This may lead to the crashing of the program. It is advised to be cautious while working with wild pointers.