<u>Answer to the question no-01</u>

a)Ans:

Ternary operator is the substitute of **if else** syntax. In this operator we have to use 3 kinds of punctuation or operators. **<** or **>** is used for making conditions ? is used for making conditions true or false, **:** is used for deciding the variable to display or put. If the condition is true then it takes the first variable or false it takes the second variable.

x>y?x:y

C Program:

```
#include <stdio.h>
int main()
{
   int x,y;
   scanf("%d%d",&x,&y);
   int a=x>y?x:y;
   printf("Large value is %d",a);
   return 0;
}
```

b)Ans:

Local Variable	**Global Variable**
1. It is declared inside of a function.	1. It is declared outside of function or next to the header file.
2. This variable works only inside of that function where it is declared.	2. This variable works in all the functions in that program.
3. If local & global variables are the same, then the local variable gets the priority to use in that function.	3. If local & global variables are the same, then if we want to use global variables then we have to use **{extern** global variable;**}** .

Answer to the question no-02

Steps:

1. Start.
2. Set variables a=0 and b=1.
3. Declare variable d.
4. Input n.
5. Print the value of a.
6. Make d=a+b.
7. Put the value of b into a.
8. Put the value of d into b and go to step no-05 if counter i<n. Else go to step no-09.
9. End.

C program:

```
#include <stdio.h>
int main()
{
   int n,i,a=0,b=1,d;
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
      if(i==n)
      {
         printf("%d",a);
      }
      else
      {
         printf("%d, ",a);
      }

      d=a+b;
      a=b;
      b=d;
   }
   return 0;
}
```

Answer to the question no-03

When we want to place the memory address of a variable in another variable, that variable is called a pointer of C. Besides, we can keep the memory address value in a variable. In this case it is called pointer to pointer.

1. Integer pointer
2. Floating pointer
3. Character pointer
4. Pointer to pointer.

Pass by Value	Pass by reference
1. Pass by value means send the value in a function.	1. Pass by reference means send the memory address of that value.
2. By passing more than one, we can only get only one value.	2. By passing reference of a value we do not need to return anything.
3. If we want to exchange value ,we have to use that variable.	3. In this case we don't have to use the variable ,only use the pointer of that value.

C Program:

Pass by Value

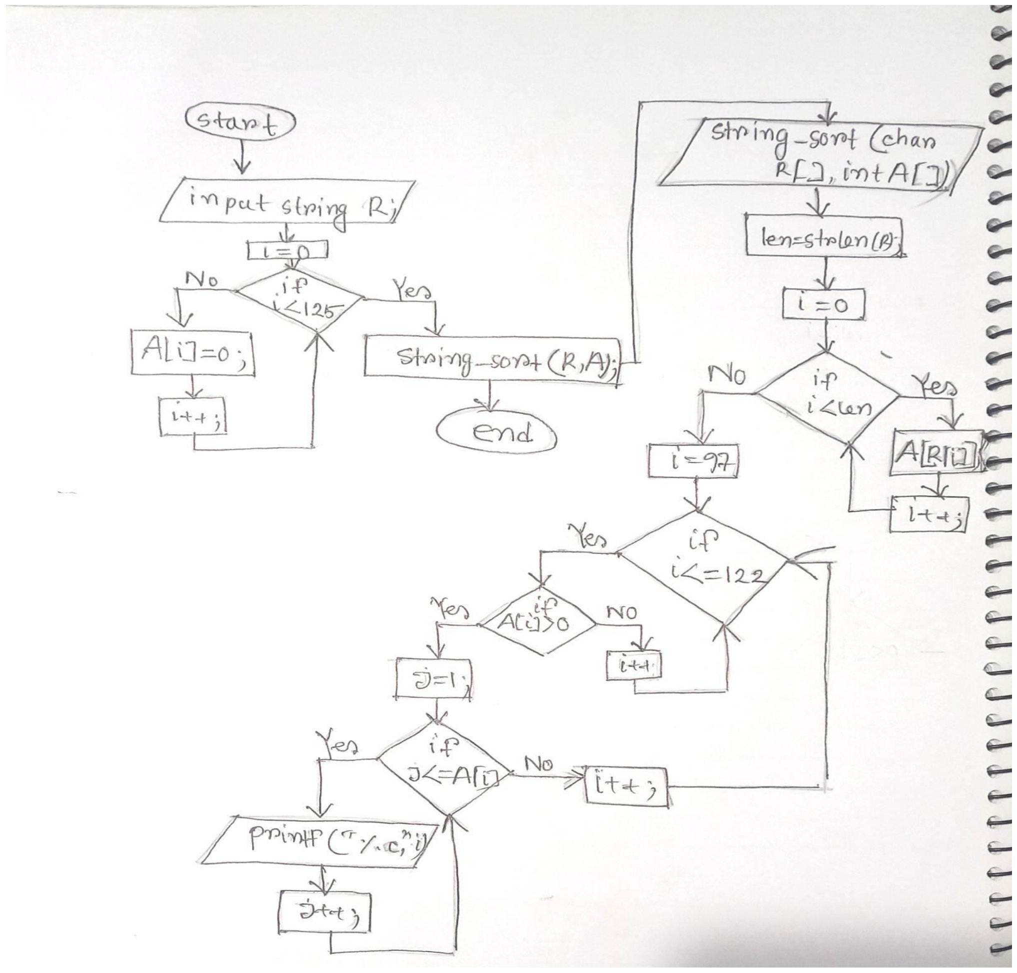
```c
#include <stdio.h>
int sum(int a,int b)
{
   int d=a+b;
   return d;
}
int main()
{
   int a,b;
   scanf("%d%d",&a,&b);
   printf("%d",sum(a,b));
   return 0;
}
```

Pass by reference

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

Answer to the question no-04

Proper Diagram Steps:



In this picture the box of A[R[i]] is written in A[R[i]]++;

C Program:

```
#include <stdio.h>
#include <string.h>
void string_sort(char R[],int A[])
{
   int len=strlen(R);
   for(int i=0;i<len;i++)
   {
      A[R[i]]++;
   }
   for(int i=97;i<=122;i++)
   {
      if(A[i]>0)
```

```
        {
           for(int j=1;j<=A[i];j++)
           {
              printf("%c",i);
           }
        }
    }
}
int main()
{
   char R[100];
   scanf("%s",R);
   int A[125];
   for(int i=0;i<125;i++)
   {
      A[i]=0;
   }
   string_sort(R,A);
   return 0;
}
```

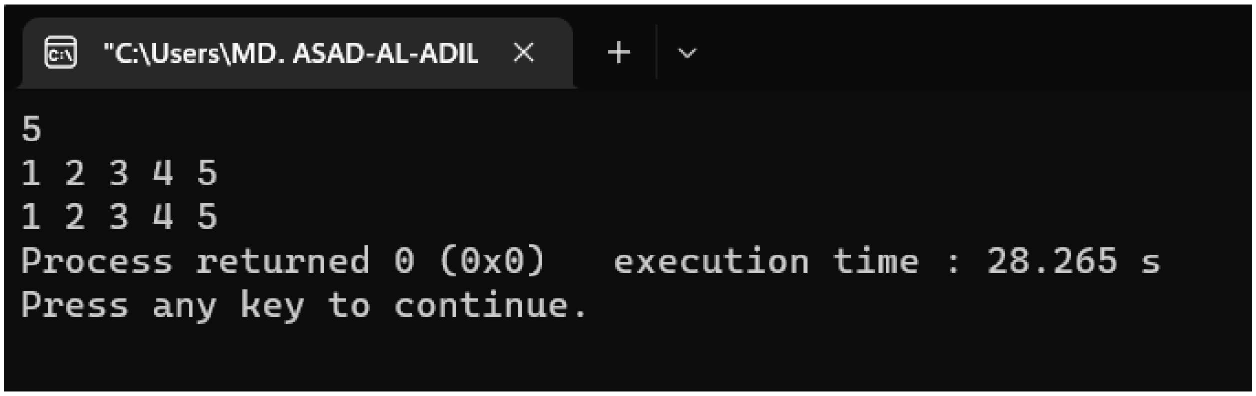
Answer to the question no-05

malloc Function	**calloc Function**
1. Full form is memory allocation.	1.Full form is contiguous allocation.
1. Having a problem with garbage value.	2. Garbage value becomes 0 when using this function.
2. We have to use '*' character in this function to create allocation.	3. We have to use ',' character in this function to create allocation.

3. example:
 	
int *Ptr;
Ptr=(int*)malloc(5*sizeof(int));
	4. example:
	
	int *Ptr;
	Ptr=(int*)calloc(5,sizeof(int));

```
#include <stdio.h>
```

```
#include <stdlib.h>
int main()
{
   int n,i;
   scanf("%d",&n);
   int *X;
   X=(int*)malloc(n*sizeof(int));
   for(i=0;i<n;i++)
   {
     scanf("%d",X+i);
   }
   for(i=0;i<n;i++)
   {
      printf("%d ",*(X+i));
   }
   return 0;
}
```



```
"C:\Users\MD. ASAD-AL-ADIL
5
1 2 3 4 5
1 2 3 4 5
Process returned 0 (0x0)   execution time : 28.265 s
Press any key to continue.
```

<u>Answer to the question no-06</u>

Function in c is a set of statements which is defined or enclosed by 2 curly brackets ({}). In a function we can do the calculation or create the logic to perform the program and make the program in a simple way. Besides, we can call that function multiple times.

User defined function are 4 types:

1. **Function with argument and return value.**

```
int sum(int a,int b)
{
   return a+b;
}
```

2. **Function with argument and no return value.**

```
 void sum(int a,int b)
{
   printf("%d",a+b);
}
```

3. **Function with no argument and return value.**

```
void sum()
{
   int a,b;
   scanf("%d%d",&a,&b);
   return a+b;
}
```

4. **Function with no argument and no return value.**

```
void sum()
{
   int a,b;
   scanf("%d%d",&a,&b);
   printf("%d",a+b);
}
```

The benefits of user defined function are:

1. We can call it multiple times as needed.
2. We can make our main function simple and clear.
3. One can easily understand the program written in function.
4. Become easy to compile.

<u>Answer to the question no-07</u>

```
#include <stdio.h>
int main()
{
   int r,c,i,j;
   scanf("%d%d",&r,&c);
   printf("\n");
   int R[r+1][c+1];
   for(i=1;i<=r;i++)
   {
      for(j=1;j<=c;j++)
```

```
        {
            scanf("%d",&R[i][j]);
        }
    }
    printf("\n");
    int a;
    for(i=1;i<=r;i++)
    {
        for(j=1;j<=c;j++)
        {
            scanf("%d",&a);
            R[i][j]=R[i][j]+a;
        }
    }
    printf("\n");
    for(i=1;i<=r;i++)
    {
        for(j=1;j<=c;j++)
        {
            printf("%d ",R[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

<u>Answer to the question no-08</u>

a)Ans:

Structure is a collection of elements that can be the same or different data type.
We use structure to avoid taking many variables. Besides, we use it to create different combined data types which we can not do in an array. It also helps to construct a complex data type which is more meaningful.

b)Ans:

There are 2 ways to access structure members.

1. By using (.) dot operator.
2. Accessing sequentially

3. Outside the curly bracket.

```
#include <stdio.h>
struct S
{
   double a;
   int x;
   char y[10];
};
int main()
{
   struct S A={22.2,25,"Adil"};
   printf("%lf %d %s",A.a,A.x,A.y);
   return 0;
}
```
	```
	#include <stdio.h>
	struct S
	{
	   double a;
	   int x;
	   char y[10];
	};
	int main()
	{
	   struct S A={.a=22.2,.x=25,.y="Adil"};
	   printf("%lf %d %s",A.a,A.x,A.y);
	   return 0;
	}
	```

```c
#include <stdio.h>
struct S
{
   double a;
   int x;
   char y[10];
};
int main()
{
   struct S A={.y="Adil"};
   A.a=22.2;
   A.x=25;
   printf("%lf %d %s",A.a,A.x,A.y);
   return 0;
}
```

We cannot assign string values with this method.

C)Ans:

```c
#include <stdio.h>
struct info
{
   int roll;
   int marks;
   char name[50];
```

```
};
int main()
{
   printf("Enter information:\n");
   struct info S;
   printf("Enter name: ");
   scanf("%s",S.name);
   printf("Enter roll number: ");
   scanf("%d",&S.roll);
   printf("Enter marks: ");
   scanf("%d",&S.marks);
   printf("\nDisplaying Information:\n");
   printf("Name: %s\n",S.name);
   printf("Roll number: %d\n",S.roll);
   printf("Marks: %d\n",S.marks);
   return 0;
```

Answer to the question no-09

5 types error of C program:

1. **Syntax error**

```
#include <stdio.h>
int main()
{
   Int a=25,b=41;
   printf("%d",a+b);
   return 0;
}
```

Here ‘i’ is in the capital.

2. **Run time error**

```
#include <stdio.h>
int main()
{
   int a=25,b=0;
   printf("%d",a/b);
   return 0;
}
```

Here a/b is an infinite value.

3. **Logical error**

```
#include <stdio.h>
int main()
{
   int a=0;
   for(int i=1;i<=10;i++;)
   {
      a=a+i;
   }
   printf("%d",a);
   return 0;
}
```

Here in for loop we use a semicolon after ‘i++’ .

4. **Semantic error**

```
#include <stdio.h>
int main()
{
   int a=10,b=25,c;
   a*b=c;
   printf("%d",c);
   return 0;
}
```

Here we use a*b=c but we have to use c=a*b;

5. **Linker error**

```
#include <stdio.h>
int Main()
{
   int a=10,b=25,c;
   c=a*b;
   printf("%d",c);
   return 0;
}
```

Here we have to write ‘main’ not ‘Main’.

Answer to the question no-10

a)Ans:

C is the first programming language and after that most compilers and most programming languages follow the syntax of C language. It is a general purpose programming language that is used for creating numerous applications. It also combines high and low level language. For that C is called the mother of all languages.

b)Ans:

Source code- It is a programming statement that is created by a person with the code editor or visual programming tool and saved in a file. The extension of this file will be which kind of file it is. For C it will .c, for C++ it will .cpp, for Python it will .py

.obj file- It is an object file. After writing the code when we run the program then we can see a black background file. When we compile , this file opens to do operations.

.exe file- It is called an executable file. It is a fully compiled version of a program that can run by the users or another person without further compilation.

c)Ans:

When I run a C program 2 types of files are created. They are:

1. Object file.
2. Application file.

Before running the program we already created the source code file of that program.

d)Ans:

In a recursion function base condition is that condition to stop the recursion function.
Real life example of recursion:
Stranger: Hey boy, where is your home?
Boy: Beside the pond.
Stranger: where is the pond?
Boy: Beside my home.
Stranger : Where is your home?
Another example is telling a lie. If a person tells a lie then he has to tell a lie again as he told a lie before. In such a way that person is calling the lie function to tell a lie.

e)Ans:

We should declare a large data type first in the structure to minimize the size of that structure and also memory optimization. Otherwise the size becomes large.