Experiment 1 Date: 21.09.2023

Advanced Use of GCC

Aim:

1. Advanced use of gcc : Important Options -o, -c, -D, -l, -I, -g, -O, -save-temps, -pg

Write a C program 'sum.c' to add two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate output using gcc command and its important options.

Program

```
#include<stdio.h>

void main(){

   int a,b;

   printf("Enter 2 numbers : ");

   scanf("%d %d",&a,&b);

   printf("Sum : %d",a+b);

}
```

GCC

GCC is a Linux-based c compiler released by the free software foundation which is usually operated via the command line. It often comes distributed freely with a Linux installation, so if you are running Unix or a Linux variant you will probably have it on your system. You can invoke gcc on a source code file simply by typing:-

gcc filename

The default executable output of gcc is "a.out", which can be run by typing"./a.out". It is also possible to specify a name for the executable file at the command line by using the syntax " -o outputfile", as shown in the following example: -

gcc filename -o outputfile

Again, you can run your program with "./outputfile". (the ./ is there to ensure to run the program for the current working directory.)

Note: if you need to use functions from the math library (generally functions from math.h" such as sin or sqrt), then you need to explicitly ask it to link with that library with the "-1" flag and the library "m":

gcc filename -o outputfile -lm

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc sum.c

mits@mits:~/Desktop/S1MCA/ADS_lab$ ./a.out sum.c

Enter 2 numbers : 10 20

Sum : 30
```

Important Options in GCC

Option: -o

To write and build output to output file.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc sum.c -o sum_out
```

Here, GCC compiles the sum.c file and generates an executable named sum_out.

Option: -c

To compile source files to object files without linking.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -c sum.c
```

This will generate an object file sum.o that can be linked separately.

Option: -D

To define a preprocessor macro.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -D debug=1 sum.c
```

This defines the macro ‘DEBUG’ with the value 1, which can be used in the source code.

Option: -l

To include a directory of header files.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -o sum.c sum_out.c -lm
```

Here, the -lm option links the math library (libm) with the sum.c.

Option: -I

To look in a directory for library files.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -o sum.c sum_out.c -I./ads_lab
```

This tells GCC to look for header files in the ads_lab directory.

Option: -g

To debug the program using GDB.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -g sum.c -o sum_out
```

This compiles sum.c with debug information, enabling you to debug the resulting executable.

Option: -O

To optimize for code size and execution time.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -O3 -o my_pgm sum.c
```

This compiles sum.c with a high level of optimization.

Option: -pg

To enable code profiling.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -pg -o my_pgm sum.c
```

This compiles source.c with profiling support, allowing you to use profilers like gprof.

Option: -save-temps

To save temporary files generated during program execution.

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -save-temps -o my_pgm sum.c
```

This will generate intermediate files, like sum.i (pre-processed source) and sum.s (assembly code), in addition to the final executable.

Experiment 2 Date: 21.09.2023

Familiarisation with GDB

Aim:

2. Familiarisation with gdb: Important Commands - break, run, next, print, display, help.

Write a C program 'mul.c' to multiply two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate sum.out which is then debug with gdb and commands.

Program

```
#include<stdio.h>

void main(){

   int a,b;

   printf("Enter 2 numbers : ");

   scanf("%d %d",&a,&b);

   printf("Product : %d",a*b);

}
```

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -g mul.c -o mul_out

mits@mits:~/Desktop/S1MCA/ADS_lab$ gdb mul_out


GNU gdb (Ubuntu 12.0.90-0ubuntu1) 12.0.90

Copyright (C) 2022 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<https://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online at:
```

```
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum1...
(gdb) run
Starting program: /home/mits/Desktop/Poojas1MCA/sum1
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter 2 numbers : 10 20
Product : 200 [Inferior 1 (process 23588) exited normally]
(gdb) quit
```

Important Commands in GDB

Command: break

Sets a breakpoint on a particular line.

Output

```
(gdb) break mul.c:5
```

Command: run

Executes the program from start to end.

Output

```
(gdb) run
```

Command: next

Executes the next line of code without diving into functions.

Output

```
(gdb) next
```

Command: print

Displays the value of a variable.

Output

```
(gdb) print a
(gdb)   a 10
```

Command: display

Displays the current values of the specified variable after every step.

Output

```
(gdb) display a

1: a=10
```

Experiment 3 Date: 29.09.2023

Familiarisation with gprof

Aim:

3. Write a program for finding the sum of two numbers using a function. Then profile the executable with gprof.

Program

```
#include<stdio.h>
int sum(int x, int y){
   return x+y;
}
void main(){
   int a,b;
   printf("Enter 2 numbers : ");
   scanf("%d %d",&a,&b);
   printf("Sum : %d",sum(a,b));
}
```

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc sum.c
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc ./a.out sum.c
Enter 2 numbers : 10 20
Sum : 30

mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -o sum.out -pg sum.c
mits@mits:~/Desktop/S1MCA/ADS_lab$ ./sum.out
Enter 2 numbers : 10 20
Sum : 30
```

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gprof ./sum.out gmon.out > pgm3.txt
```

pgm3.txt

```
Flat profile:

Each sample counts as 0.01 seconds.

 no time accumulated

  %   cumulative   self              self     total

 time   seconds   seconds    calls  Ts/call  Ts/call  name

  0.00      0.00     0.00        1     0.00     0.00  sum
```

Experiment 4 Date: 29.09.2023

Different types of functions

Aim:

4. Write a program for finding the sum of two numbers using different types of functions.

Algorithm:

Void main()

```
1.    Start
2.    Declare ch,a,b.
3.    Display choices.
4.    Read option ch.
a.    if ch==1 call sum1().
b.    if ch==2 input a and b and call sum2().
c.    if ch==3 print sum3().
d.    if ch==3 input a and b and print sum4().
5.    Repeat steps 3 while ch>0&&ch<4.
6.    Stop.
      void sum1()
1.    Start
2.    Declare a and b.
3.    Read a and b.
4.    Print a+b.
5.    Exit.
      void sum2(int a, int b)
1.    Start
2.    Print a+b.
3.    Exit.
      int sum3()
1.    Start
2.    Declare a and b.
```

```
3.      Read a and b.
4.      Return a+b.
5.      Exit.

        int sum4(int a, int b)
1.      Start
2.      Return a+b
3.      Exit.
```

Program

```
#include<stdio.h>
void sum1(){
   int a,b;
   printf("Enter 2 numbers : ");
   scanf("%d %d",&a,&b);
   printf("Sum : %d",a+b);
}
void sum2(int a, int b){
   printf("Sum : %d",a+b);
}
int sum3(){
   int a,b;
   printf("Enter 2 numbers : ");
   scanf("%d %d",&a,&b);
   return a+b;
}
int sum4(int a, int b){
   return a+b;
}
void main(){
   int ch,a,b;
   do{
        printf("1. Function without return type and arguments\n2. Function without return type and with arguments\n3. Function with return type and without arguments\n4. Function with return type and arguments\n5. Exit\nEnter your choice(1-4): ");

        scanf("%d", &ch);

        switch(ch){

            case 1: sum1();

                    break;

            case 2: printf("Enter 2 numbers : ");

                    scanf("%d %d",&a,&b);

                    sum2(a,b);

                    break;


            case 3: printf("Sum : %d",sum3());

                    break;

            case 4: printf("Enter 2 numbers : ");

                    scanf("%d %d",&a,&b);

                    printf("Sum : %d",sum4(a,b));

                    break;

        }

    }while(ch>0&&ch<4);

}
```

Output

```
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc PGM1.c

mits@mits:~/Desktop/S1MCA/ADS_lab$ ./a.out PGM1.c


1. Function without return type and arguments

2. Function without return type and with arguments

3. Function with return type and without arguments

4. Function with return type and arguments

5. Exit

Enter your choice: 1
Enter 2 numbers : 10 20
Sum : 30

1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 2
Enter 2 numbers : 25 25
Sum : 50

1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 3
Enter 2 numbers : 100 100
Sum : 200

1. Function without return type and arguments
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 4
Enter 2 numbers : 250 250
Sum : 500

1. Function without return type and arguments
```

```
2. Function without return type and with arguments
3. Function with return type and without arguments
4. Function with return type and arguments
5. Exit
Enter your choice: 5

mits@mits:~/Desktop/S1MCA/ADS_lab$
```

Experiment 5 **Date: 06.10.2023**

Array Operations

Aim:

To implement a menu driven program to perform following array operations

i. Insert an element to a particular location
ii. Delete an element from a particular location
iii. Traverse

Algorithm:

Void main

```
1.Start
2.Declare the variables  i, a[10], n, ch=1,r
3.create a array create a menu driven to insert delete and traverse
4. stop
```

insert()

```
1. start
2.declare variables item, pos, i;
3. check the condition if(n<10)
     print enter the element to insert
     print Enter the position to insert
 4.check the loop  for(i=n-1; i>=pos; i--)
      Put a[i+1]=a[i], then  a[pos]=item;
      n=n+1;
 5. check the condition for(i=0;i<n;i++)
     Print a[i]
 6. else
     Print Not possible to insert, OVERFLOW
7. stop
```

delete()

```
1.start
2. declare the variables int pos,item,i;
3. Print Enter the position to delete
4. Set item=a[pos];
5. Check the condition for(i=pos;i<n-1;i++)
       Set a[i]=a[i+1], Then n=n-1;
6. Check another condition for(i=0;i<n;i++)
    Print a[i]
7.stop
```

traverse()

```
1.print Array
3. print a[i] when the condition
```

For(i=0;i<n;i++)

Program

```
#include<stdio.h>
#include<stdlib.h>
int p,x;
void insert(int *a,int n)
{
        int item,pos,i;
        if(n<10)
        {
        printf(" enter the element to insert:\n");
        scanf("%d", &item);
        printf("Enter the position to insert: ");
        scanf("%d",&pos);
        for(i=n-1;i>=pos;i--)
        a[i+1]=a[i];
        a[pos]=item;
        n=n+1;
        for(i=0;i<n;i++)
printf("%d\n",a[i]);
        }
        else
        printf("Not possible to insert, OVERFLOW\n");
}
void del(int *a,int n)
{
        int pos,item,i;
        printf("Enter the position to delete: ");
        scanf("%d",&pos);
        item=a[pos];
        for(i=pos;i<n-1;i++)
        a[i]=a[i+1];
        n=n-1;
        for(i=0;i<n;i++)
printf("%d\n",a[i]);
}
void traversal(int *a,int n)
{
printf("Array:");
for(int i=0;i<n;i++)
printf("\n%d ",a[i]);
}
void main()
{
int i,a[10],n,ch=1;
char r;
printf("Enter the size of the array : ");
scanf("%d", &n);
printf("Enter the array : ");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
do
{
int ch;
printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice : ");
scanf("%d",&ch);
switch(ch){
case 1: insert(a,n);
        break;
case 2: del(a,n);
        break;
case 3: traversal(a,n);
break;
case 4:
ch=0;
printf("Thank you!");

exit(0);
default:
printf("Invalid choice!!!");
break;
}
//printf("Do you want to continue\n");
//scanf("%c",&r);
}while(ch==1);
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc menudriven.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
Enter the size of the array : 5
Enter the array : 3
4
5
6
7

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice : 1
 enter the element to insert:
6
Enter the position to insert: 3
3
```

```
4
5
6
6
7

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice : 2
Enter the position to delete: 5
3
4
5
6

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice : 3
Array:
3
4
5
6
6
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice : 4
Thank you!mits@mits-Lenovo-S510:~/Desktop/s1mca
```

Experiment 6 **Date: 06.10.2023**

Sort Array

Aim

Program to sort an integer array

Algorithm:

Void main()

```
1.  Start
2.  declare and initialise the variable
      a[10], n, ans, i, j, temp
3.  print Enter the number of elements
4.  Print Enter the elements
5.  Check the condition for(i=0; i<n; i++)
6.  Checking another loop  for(i=0;i<n-1;i++)
7.  Check the another loop for(j=0;j<n-i-1;j++
8.          if(a[j]>a[j+1])
9.          temp=a[j];
10.         a[j]=a[j+1];
11.         a[j+1]=temp
12. Call the bubble sort
      b_sort(a,n)
13. Stop
```

Bubble sort()

```
1. Start
2. Declare and initialize variables i,temp,j
3. Check the condition for(i=0;i<n-1;i++
4. Check another condition
   for(j=0;j<n-i-1;j++)
5. Check with if statement
6. if(a[j]>a[j+1])

       temp=a[j];

       a[j]=a[j+1];

       a[j+1]=temp;

7. check the condition   for (i=0;i<n;i++)

   print a[i]

8. stop
```

Program

```
#include<stdio.h>
void b_sort(int *a,int n)
{
  int i,temp,j;
  for(i=0;i<n-1;i++)
  {
          for(j=0;j<n-i-1;j++)
          {
          if(a[j]>a[j+1])
          {
          temp=a[j];
          a[j]=a[j+1];
          a[j+1]=temp;
          }
          }
  }
  for(i=0;i<n;i++)
          printf("%d\t",a[i]);
}
void main()
{
  int a[10],n,ans,i,j,temp;
  printf("Enter the number of elements :");
  scanf("%d",&n);
  printf("Enter the elements :");
  for(i=0;i<n;i++)
          scanf("%d",&a[i]);
  for(i=0;i<n-1;i++)
  {
          for(j=0;j<n-i-1;j++)
```

```
            {
            if(a[j]>a[j+1])
            {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
            }
            }
   }
   b_sort(a,n);
 }
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc sort1.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
Enter the number of elements :5
Enter the elements :1
4
7
8
3
1          3    4       7       8       mits@mits-Lenovo-S510:~/Desktop/s1mca
```

Experiment 7 **Date: 06.10.2023**

Searching Operations

Aim:

Program to implement linear search and binary search

Algorithm:

void sort(int arr[],int n)

```
1. Start
2. for(int i=0;iarr[j])
{
int a=arr[i];
arr[i]=arr[j];
arr[j]=a;
}
3. Stop.
```

void linearsearch(int arr[],int n,int ele)

```
1. Start
2. for(int i=0;i<n;i++)
if(arr[i]==ele)
{
Print element found at position
}
3. Stop
```

void binarysearch(int arr[],int n,int ele)

```
1. Start
2. int lb=0, ub=9, mid=(lb+ub)/2;
3. if(arr[mid]==ele)
     print Element found
4. else if(ele>arr[mid])
       lb=mid+1;
5. else ub=mid-1;
6. Stop
```

main()

```
1. Start
2. Declare the array size of the array and Function
3.declare the variables i,a,ch,ele;
4. . Print the array
5.Enter the menudriven program
6. Enter the search option
```

```
Case 1:
Call linearsearch(arr,n,ele);
break;
Case 2:
Call sort(arr,n);
Enter search element
Call binarysearch(arr,n,ele);
break;
Default:
Print invalid option
break;
7. Stop.
```

Program

```
#include <stdio.h>
void main()
 {
void sort(int arr[],int n);
void linearsearch(int arr[],int n,int ele);
void binarysearch(int arr[],int n,int ele);
int i,a,ch,ele;
int arr[10]={5,9,8,6,3,0,12,65,1,90};
int n=10;
printf("The array is:");
for(i=0;i<n;i++)
 {
printf("%d\t",arr[i]);
 }
printf("\nSearch option:\n1)Linear search\n2)Binary search");
while(ch<3)
 {
printf("Enter the option:");
scanf("%d",&ch);
switch(ch)
 {
case 1:
 {
printf("\nEnter the element to search:");
scanf("%d",&ele);
linearsearch(arr,n,ele);
break;
}
case 2:
{
sort(arr,n);
printf("The sorted array is:");
for(i=0;i<n;i++){
printf("%d\t",arr[i]);
}
printf("\nEnter the element to search:");
scanf("%d",&ele);
binarysearch(arr,n,ele);
break;
}
default:
{
printf("invalid option");
break;
}
}
}
}
void sort(int arr[],int n) {
for(int i=0;i<n;i++)
{
for(int j=i+1;j<n;j++)
{
if(arr[i]>arr[j])
{
int a=arr[i];
arr[i]=arr[j];
arr[j]=a;
}
}
}
}
void linearsearch(int arr[],int n,int ele)
{
for(int i=0;i<n;i++)
{
```

```
if(arr[i]==ele)
 {
printf("Element %d found at %d th position",ele,i);
break;
 }
 }
 }
void binarysearch(int arr[],int n,int ele)
 {
int lb=0;
int ub=9;
int mid;
do{
mid=(lb+ub)/2;
if(arr[mid]==ele)
 {
printf("%d Present at %d th position",ele,mid);
 }
else if(ele>arr[mid]){
lb=mid+1;
 }
else
 {
ub=mid-1;}
 }while(arr[mid]!=ele);
 }
```

Output

```
mits@mits-HP-280-Pro-G6-Microtower-PC:~$ cd Desktop
mits@mits-HP-280-Pro-G6-Microtower-PC:~/Desktop$ cd s1mca mits@mits-
HP-280-Pro-G6-Microtower-PC:~/Desktop/s1mca$ gcc search.c mits@mits-HP-
280-Pro-G6-Microtower-PC:~/Desktop/s1mca$ ./a.out The array is: 5 9 8 6 3 0 12
65 1 90
 Search option:
 1) Linear search
2) Binary search

Enter the option:1
Enter the element to search:6
```

```
Element 6 found at 3 th position

Enter the option: 2
The sorted array is:0 1 3 5 6 8 9 12 65 90
Enter the element to search: 65
65 Present at 8 th position
```

Experiment 8 Date: 08.10.2023

Matrix Operations

Aim:

Perform addition, subtraction and multiplication of two matrices using switch.

Algorithm:

Int main()

1. start
2. declare and initialize the two dimensional array
3. int a[][3] = { {5,6,7}, {8,9,10}, {3,1,2} };
   int b[][3] = { {1,2,3}, {4,5,6}, {7,8,9} }
   int c[3][3];
4. Print First Matrix
5. Print Second Matrix
6. create a menudriven to matrix operation
   - addition
   - subtraction
   - multiplication

addition()

1. create a function for addition
2. void add(int m[3][3], int n[3][3], int sum[3][3])
3. giving for loop for the condition

```
FOR (I=0 ; I<N ; I++)
   FOR(J=0 ; J<3 ; J++)
   SUM[I][J] =M[I][J] + N[I][J]
```

subtraction()

1. create a function for subtraction
2. void subtract(int m[3][3], int n[3][3], int result[3][3])
3. declare two dimensional array
4. giving for loop for the condition

```
FOR (I=0 ; I<N ; I++)
   FOR(J=0 ; J<3 ; J++)
   RESULT[I][J] =M[I][J] - N[I][J]
```

multiplication()

1. create a function for subtraction
2. void multiply(int m[3][3], int n[3][3], int result[3][3])
3. giving for loop for the condition

```
        FOR (I=0 ; I<N ; I++)
           FOR(J=0 ; J<3 ; J++)
           RESULT[I][J] =M[I][J] * N[I][J]
```

Program

```
#include<stdio.h>
#include<stdlib.h>
void add(int m[3][3], int n[3][3], int sum[3][3])
{
  for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
        sum[i][j] = m[i][j] + n[i][j];
}
void subtract(int m[3][3], int n[3][3], int result[3][3])
{
  for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
        result[i][j] = m[i][j] - n[i][j];
}
void multiply(int m[3][3], int n[3][3], int result[3][3])
{
  for(int i=0; i < 3; i++)
  {
        for(int j=0; j < 3; j++)
        {
        result[i][j] = 0; // assign 0
        // find product
        for (int k = 0; k < 3; k++)
        result[i][j] += m[i][k] * n[k][j];
        }
  }
```

```
}
void display(int matrix[3][3])
{
  for(int i=0; i<3; i++)
  {
            for(int j=0; j<3; j++)
            printf("%d\t",matrix[i][j]);

            printf("\n"); // new line
  }
}
int main()
{
  int a[][3] = { {5,6,7}, {8,9,10}, {3,1,2} };
  int b[][3] = { {1,2,3}, {4,5,6}, {7,8,9} };
  int c[3][3];
  printf("First Matrix:\n");
  display(a);
  printf("Second Matrix:\n");
  display(b);
  int choice;
  do
  {
            printf("\nChoose the matrix operation,\n");
            printf("1. Addition\n");
            printf("2. Subtraction\n");
            printf("3. Multiplication\n");
            printf("4. Exit\n");
            printf("Enter your choice: ");
            scanf("%d", &choice);
            switch (choice) {
```

```
            case 1:
            add(a, b, c);
            printf("Sum of matrix: \n");
            display(c);
            break;
            case 2:
            subtract(a, b, c);
            printf("Subtraction of matrix: \n");
            display(c);
            break;
            case 3:
            multiply(a, b, c);
            printf("Multiplication of matrix: \n");
            display(c);
            break;
            case 4:
            printf("Thank You.\n");
            exit(0);
            default:
            printf("Invalid input.\n");
            printf("Please enter the correct input.\n");
            }
  }while(1);
  return 0;
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc matrix.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
First Matrix:
5    6    7
8    9    10
3    1    2
Second Matrix:
```

```
1    2    3
4    5    6
7    8    9
Choose the matrix operation,
1. Addition
2. Subtraction
3. Multiplication
4. Exit
Enter your choice: 1
Sum of matrix:
6    8    10
12    14    16
10    9    11
Choose the matrix operation,
1. Addition
2. Subtraction
3. Multiplication
4. Exit
Enter your choice: 2
Subtraction of matrix:
4    4    4
4    4    4
-4    -7    -7
Choose the matrix operation,
1. Addition
2. Subtraction
3. Multiplication
4. Exit
Enter your choice: 3
Multiplication of matrix:
78    96    114
114    141    168
21    27    33

Choose the matrix operation,
1. Addition
2. Subtraction
3. Multiplication
4. Exit
Enter your choice: 4
Thank You.
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 9 **Date: 08.10.2023**

Stack Operations

Aim:

Program to implement stack operations using arrays.

<u>Algorithm:</u>

Int main()

```
1.  start
2.  declare and initialize the variables
3.  stack[100],ch,n,top,item,i;
4.   call the function void push(void);
5.  Call the function void pop(void);
6.  Call the function void display(void);
7.  top=-1;
8.  Print  Enter the size of stack
9.  create a mendriven program to perform stack operation
10. stop
```

push()

```
1.  start
2.  create a function pop()
3.  if(top>=n-1),
4.  the stack is overflow
5.  else
6.  print Enter a value to be pushed
7.  top++;
8.  stack[top]=item;
9.  stop
```

pop ()

```
1.  start
2.  create afunction pop()
3.  if(top<=-1)
```

```
4.  print  Stack is under flow
5.  else
6.  print The popped elements is , stack[top]
7.  top—
8.  stop
```

display()

```
1.start
 2.  create a function to display()
 3.check condition if(top>=0)
  4.   Print  The elements in stack
  5.    check condition for(i=top; i>=0; i--)
       Print stack[i]
  6.   else
        print  The STACK is empty
  7.   stop
```

Program

```
#include<stdio.h>

#include<stdlib.h>

int stack[100],ch,n,top,item,i;

void push(void);

void pop(void);

void display(void);

int main()

{

        top=-1;

        printf("\n Enter the size of stack:");

        scanf("%d",&n);
```

```
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
do
{
printf("\n Enter the Choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
{
        push();
        break;
}
case 2:
{
        pop();
        break;
}
case 3:
{
        display();
        break;
}
case 4:
{
exit(0);
        break;
}
default:
{
        printf ("\n\t Please Enter a Valid Choice");
}
```

```
        }
        }
        while(ch!=4);
        return 0;
}
void push()
{
        if(top>=n-1)
        {
        printf("\n\tSTACK is over flow");

        }
        else
        {
        printf(" Enter a value to be pushed:");
        scanf("%d",&item);
        top++;
        stack[top]=item;
        }
}
void pop()
{
        if(top<=-1)
        {
        printf("\n\t Stack is under flow");
        }
        else
        {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
        }
}
void display()
{
        if(top>=0)
        {
        printf("\n The elements in stack \n");
        for(i=top; i>=0; i--)
        printf("\n%d",stack[i]);


        }
        else
        {
        printf("\n The STACK is empty");
        }
}
```

Output

```
its@mits-Lenovo-S510:~/Desktop/s1mca$ gcc stack1.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
 Enter the size of stack:5
    1.PUSH
    2.POP
    3.DISPLAY
    4.EXIT
 Enter the Choice:1
 Enter a value to be pushed:3
 Enter the Choice:3
 The elements in stack  3
 Enter the Choice:1
 Enter a value to be pushed:4
 Enter the Choice:2
 The popped elements is 4
 Enter the Choice:3
 The elements in stack 3
 Enter the Choice:4
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```