Experiment 1 **Date: 08-09-2025**

Advanced Use of GCC

Aim:

Create appropriate C programs. Compile and generate output using gcc command and its important options-o, -c, -D, -I, -g, -O, -save-temps

About GCC Compiler

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
```

Program 1

Q.write a program to add two numbers

```
#include<stdio.h>
void main()
{
int a,b,sum;
printf("Enter a number:");
scanf("%d",&a);
printf("Enter another number:");
scanf("%d",&b);
sum=a+b;
printf("Sum is:%d\n",sum);
}
```

<u>Output</u>

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc pgm3.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a.out
Enter a number:5
Enter another number:4
Sum is:9
```

Important Options in GCC

Option: -o

To write and build output to output file.

<u>Output</u>

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc pgm3.c -o sum_out
```

Here, GCC compiles the pgm3.c file and generates an executable named sum_out.

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./sum_out
Enter a number:10
Enter another number:20
Sum is:30
```

Option: -save-temps

To save temporary files generated during program execution.

<u>Output</u>

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc -save-temps -o my_pgm pgm3.c
```

This will generate intermediate files, like pgm3.i (pre-processed source) and pgm3.s (assembly code), in addition to the final executable.

Option: -g

gcc -g generates debug information to be used by GDB debugger.

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc -g sum.c -o sum_out

mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gdb sum_out

GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1

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 sum_out...

(gdb) run

Starting program: /home/mits/Faseeh/DS/sum_out

[Thread debugging using libthread_db enabled]

Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Enter First Number:5

Enter Second Number:2

The Sum =7[Inferior 1 (process 20970) exited normally]

(gdb) quit
```

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

Option: -c

gcc -c compiles source files without linking. When we compile a program, the ‘C’ compiler will generate object files “*.o” files.* After that linker will generate a “.out” file. That means, it is a two steps process; one step is to compile the program and another step is to link the object files and generate the executable file.

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc -c sum.c

mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc sum.o -o a_out

mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a_out

Enter First Number:5

Enter Second Number:2

The Sum =7
```

This will generate an object file pgm3.o that can be linked separately. This is the way to create an object file and use the functions in object files from different code modules.

Program 2

```
/* Write a program with preprocessor directives. #ifdef is used to include a section
of code if a certain macro is defined by #define.*/
// myfile.c
#include <stdio.h>
void main()
{
#ifdef SAMPLE
        printf("With preprocessor directive= %d\n",SAMPLE);
#else
        printf("With out #ifdef\n");
#endif
}
```

Important Option in GCC

Option: -D

gcc -D defines a macro to be used by the preprocessor.

Output

a) Build myfile.c and run it with the macro, SAMPLE defined:

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc myfile.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc -D SAMPLE
myfile.c -o myfile
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./myfile
With preprocessor directive= 1
```

b) Build myfile.c and run it without the macro, SAMPLE defined:

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc myfile.c -o myfile
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./myfile
With out #ifdef
```

Program 3

```
/* Create a program with macro and saved this as header file. Create another C
program which include this header file. */

// myheader.h
#define NUM1 5
```

save this file as **src/myheader.h**

```
// myfile.c
#include <stdio.h>
#include "myheader.h"

void main()
{
        int num = NUM1;
        printf("num=%d\n", num);
}
```

save this file as myfile.c

Important Option in GCC

Option: -I

gcc -I include directory of header files. This flag is used to specify additional directories where header files are located. It helps the preprocessor find the necessary headers when compiling the code.

Output

a) Build *myfile.c* without include directory *src* :

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc myfile2.c
myfile2.c:2:18: fatal error: myheader.h: No such file or directory
    2 |        #include "myheader.h"
      |                 ^~~~~~~~~~~~
compilation terminated.
```

b) Build *myfile.c* with include directory *src* :

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc -Isrc myfile2.c -o
   myfile2_out

mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./myfile2_out

num=5
```

Experiment 2 **Date: 12-09-2025**

Familiarisation with GDB

Aim:

Write a C program ‘multiply.c’ to multipy two numbers. Read the input from Standard Input and write output to Standard output. Compile and generate output which is then debug with gdb and use the important commands break, run, next, print, display

Program

```
#include<stdio.h>
void main()
{
        int a,b,mul;
        printf("Enter your first number: ");
        scanf("%d",&a);
        printf("Enter your second number: ");
        scanf("%d",&b);
        mul=a*b;
        printf("product is: %d",mul);
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc multiply.c -o multiply_out
mits@mits-Veriton-M200-H510:~/Faseeh/DS$./multiply_out
Enter your first number: 3
Enter your second number: 4
product is: 12
```

Important Commands in GDB

option -g

gcc -g generates debug information to be used by GDB debugger.

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ -g multiply.c -o multiply_out
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gdb multiply_out
```

Output

```
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
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 multiply_out...
(gdb)
```

a. Command: run

Executes the program from start to end.

Output

```
(gdb) run
Starting program: /home/mits/Faseeh/DS/multiply_out
[Thread debugging using libthread_db enabled]
```

```
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter First Number:5
Enter Second Number:3
Result=15[Inferior 1 (process 5931) exited normally]
```

b. Command: l - for Display the code

Type “l” at gdb prompt to display the code.

Output

```
(gdb) l
1       #include <stdio.h>
2       int main()
3       {
4       int a,b,mul;
5       printf("Enter two numbers:");
6       scanf("%d %d",&a,&b);
7       mul=a*b;
8       printf("product is:%d",mul);
9       return 0;
10      }
```

c. Command: break

Sets a breakpoint on a particular line.

Output

```
(gdb) break multiply.c:4
Breakpoint 1 at 0x5555555551a4: file multiply.c, line 5.
(gdb) run
Starting program: /home/mits/Faseeh/DS/multiply_out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Breakpoint 1, main () at multiply.c:5
5 printf("Enter two numbers:");
```

d. Command: next

Executes the next line of code without diving into functions.

Output

```
(gdb) next
6         scanf("%d%d",&a,&b);
(gdb) next
Enter two numebrs:4 5
7          mul=a*b;
```

e. Command: clear

Clears all breakpoints.

Output

```
(gdb) clear
No breakpoint at this line.
```

f. Command: print

Displays the value of a variable.

Output

```
(gdb) print a
(gdb)$1 = 2
```

g. Command: display

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

Output

```
(gdb) display a
1: a = 2
```

h. Command: quit

Exits out of GDB.

```
(gdb) quit
```

Experiment 3 Date:12-09-2025

Familiarisation with gprof

Aim:

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

gprof

Gprof is a profiling program which collects and arranges statistics on our programs. Basically, it looks into each of our functions and inserts code at the head and tail of each one to collect timing information. Then, when we run our program normally, it creates "gmon.out". In order to produce profiling information, the program must be compiled with specific options to tell it to record this information. The important options are: ‘-pg' (profile graph) and '-g' (debug information) must be included.

Program

```
#include<stdio.h>
int sum1(int a,int b);
void main()
{
        int num1,num2,x;
        printf("Enter two numbers:");
        scanf("%d%d",&num1,&num2);
        x=sum1(num1,num2);
        printf("sum is %d\n",x);
}
int sum1(int a,int b)
{
        int s=a+b;
        return s;
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc sum.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$./a.out
Enter two numbers:10 25
sum is 35
```

Option: -pg

To compile a source file for profiling, specify the '-pg' option when we run the compiler. -pg', alters either compilation or linking to do what is necessary for profiling.

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$  gcc -pg -o sum.out sum.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$  ./sum.out
Enter two numbers : 10 20
sum is 30

mits@mits-Veriton-M200-H510:~/Faseeh/DS$  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:12-09-2025

Different types of functions

Aim:

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

a) Function with argument and return type

b) Function with argument and no return type

c) Function without argument and return type

d) Function without argument and no return type

This program should have menu driven options.

Algorithm:

main()

```
step 1: start
step 2: display
        1. function with argument and return type
        2. function with argument and no return type
        3. function without argument and return type
        4. function without argument and no return type
        5. exit
step 3: repeat until while choice!=5
step 4 : read option into choice
step 5 : switch
        case 1:
                read a,b
                call function sum1(a,b)
                print result
        case 2:
                read a,b
                call function sum2(a,b)
        case 3:
                call function sum3()
```

```
                    print result
            case 4:
                    call function sum4()
            case 5:
                    exit
            default:
                    invalid Choice
    step 6: stop
```

int sum1(int x, int y)

```
    step 1: start
    step 2: return x+y
    step 3 :exit
```

void sum2(int x, int y)

```
    step 1: start
    step 2: print x+y
    step 3: exit
```

int sum3()

```
    step 1: start
    step 2: read two numbers, a, b
    step 3: return a+b
```

void sum4()

```
    step 1: start
    step 2: read two numbers, a, b
    step 3: print a+b
    step 4: exit
```

Program

```
#include <stdio.h>
int sum1(int x,int y);
void sum2(int x,int y);
int sum3();
void sum4();
void main()
{
        int a,b,c,d,result1,result2,ch=0;
        printf("1.Function with argument and return type\n2.Function with argument and
        no return type\n3.Function without argument and return type\n4.Function without
        argument and no return type\n5.Exit\n");
        while(ch!=5)
        {
                printf("\nEnter your choice (1-5) : ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("Enter two numbers: ");
                                scanf("%d%d",&a,&b);
                                result1=sum1(a,b);
                                printf("Sum = %d\n",result1);
                                break;
                        case 2:
                                printf("Enter two numbers: ");
                                scanf("%d%d",&a,&b);
                                sum2(a,b);
                                break;

                        case 3:
```

```
                        result2=sum3();
                        printf("Sum = %d\n",result2);
                        break;
                    case 4:
                         sum4();
                        break;
                    case 5:
                        printf("Exit");
                        break;
                    default: printf("Enter a valid choice!!!!\n");
                        break;
                }
        }
}

int sum1(int x,int y)
{
        return x+y;
}

void sum2(int x,int y)
{
        printf("Sum = %d\n",x+y);
}

int sum3()
{
        int a,b;
        printf("Enter two numbers: ");
        scanf("%d%d",&a,&b);
        return a+b;
}
```

```
void sum4()
{
        int a,b;
        printf("Enter two numbers :");
        scanf("%d%d",&a,&b);
        printf("Sum = %d\n",a+b);
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc sum.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a.out
1.Function with argument and return type
2.Function with argument and no return type
3.Function without argument and return type
4.Function without argument and no return type
5.Exit
Enter your choice (1-5) : 1
Enter two numbers: 10 20
Sum = 30
Enter your choice (1-5) : 2
Enter two numbers: 5 10
Sum = 15
Enter your choice (1-5) : 3
Enter two numbers: 20 30
Sum = 50
Enter your choice (1-5) : 0
Enter a valid choice!!!!
Enter your choice (1-5) : 4
Enter two numbers: 5 5

Sum = 10

Enter your choice (1-5) : 5

Exit
```

Experiment 5 **Date:15-09-2025**

Array Operations

Aim:

To implement a menu driven program to perform following array operations using function

i. Insert an element to a particular location

ii. Delete an element from a particular location

iii. Traverse

Algorithm:

main()

```
step 1: start
step 2: declare i, n, arr[20], ch=0, size=20
step 3: read size of array, n
step 4: read for i=0 to n do
            read arr[i]
step 5: display
            1. Insert an element
            2. Delete an element
            3. Traverse
            4. Exit
step 6: while ch!=4 do
step 7: read user's choice, ch
step 8: switch
        case 1:
                n= call insert(n, arr, size)
        case 2:
                n= call delete(n, arr)
        case 3:
```

```
                    call traverse(n, arr)
            case 4:
                    exit
            default:
                    invalid Input
    step 9: stop
```

int insert(int n, int arr[], int size)

```
    step 1: start
    step 2: declare item, k, j, i
    step 3: read number to be inserted, item
    step 4: read position where you want to insert, k
    step 5: if size == n
                display overflow
                go to step 13
    step 6: set j=n-1
    step 7: repeat until j>=k
                set arr[j+1] = arr[j]
                set j=j-1
    step 8: set arr[k] = item
    step 9: set n = n+1
    step 10: display “Array after insertion”
    step 11: for i = 0 to n do
                print arr[i]
    step 12: return n
    step 13: exit
```

int delete(int n, int arr[])

```
    step 1: start
    step 2: declare item, j, k, i
```

```
step 3: if n == 0
            print underflow
            go to step 10
step 4: read position of the element to be deleted, k
step 5: set item = arr[k]
step 6: for j=k to n-1 do
            set arr[j] = arr[j+1]
step 7: set n= n-1
step 8: if n == 0
            print array after deletion = NULL
        else
            print array after deletion:
            for i=0 to n do
                print arr[i]
step 9: return n
step 10: exit
```

int traverse(int n, int arr[])

```
step 1: start
step 2: if n == 0
            print Array is empty
        else
            print Elements are
            for i=0 to n do
                print arr[i]
step 3: exit
```

Program

```
#include <stdio.h>
int insert(int n,int arr[],int size);
int delete(int n,int arr[]);
void traverse(int n,int arr[]);
void main()
{
        int arr[20],size=20,ch=0,n,i;
        printf("Enter the size of the array: ");
        scanf("%d",&n);
        printf("Enter %d elements: ",n);
        for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
        printf("\n1. Insert an Element\n2. Delete an Element\n3. Traverse\n4. Exit\n");
        while(ch!=4)
        {
                printf("\nEnter your choice: ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                 n=insert(n,arr,size);
                                break;
                        case 2:
                                n=delete(n,arr);
                                break;
                        case 3:
                                traverse(n,arr);
                                break;
                        case 4:
                                printf("Exit\n");
                                break;
                        default:
                                printf("Enter a valid choice!!");


                }
        }
}
int insert(int n,int arr[],int size)
{
        int item,k,j,i;
        if(size==n)
        {
                printf("Overflow\n");
        }
        printf("Enter the number to be inserted: ");
        scanf("%d",&item);
        printf("Enter the position where you want to insert: ");
        scanf("%d",&k);
        for(j=n-1;j>=k;j--)
        {
                arr[j+1]=arr[j];
        }
        arr[k]=item;
        n=n+1;
        printf("\nArray after insertion: ");
        for(i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
        }
        printf("\n");
        return n;
}

int delete(int n,int arr[])
{
        int k,item,j,i;
        if(n==0)
        {
                printf("Underflow");
        }
        printf("Enter the position you want to delete: ");
        scanf("%d",&k);
        item=arr[k];
        for(j=k;j<n-1;j++)
        {
                arr[j]=arr[j+1];
        }
        n=n-1;
        if(n==0)
        {
                printf("\nArray after Deletion = Null");
                printf("\n");
        }
        else
        {
                printf("\nArray after Deletion: ");
                for(i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
```

```
          }
                    printf("\n");
          }
          return n;
}
void traverse(int n,int arr[])
{
          if(n==0)
          {
                    printf("\nArray is empty");
                    printf("\n");
          }
          else
          {
                    printf("\nElements are : ");
                    for(int i=0;i<n;i++)
                    {
                              printf("%d ",arr[i]);
                    }
          }
          printf("\n");
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc array.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a.out
Enter the size of the array: 3
Enter 3 elements: 10
20
```

```
30
1. Insert an Element
2. Delete an Element
3. Traverse
4. Exit
Enter your choice: 1
Enter the number to be inserted: 50
Enter the position where you want to insert: 0
Array after insertion: 50 10 20 30
Enter your choice: 3
Elements are : 50 10 20 30
Enter your choice: 2
Enter the position you want to delete: 0
Array after Deletion: 10 20 30
Enter your choice: 4
Exit
```

Experiment 6 **Date:17-09-2025**

Array Sorting

Aim:

Program to sort an integer array

Algorithm:

```
step 1: start
step 2: declare arr[20],n,i,j,temp
step 3: read the size of the array
step 4: for each i = 0 to n
            read the array elements,arr[i]
step 5: for each i= 0 to n
            print the array elements before sorting,arr[i]
step 6: for each i = 0 to n-1
            for each j = 0 to n-i-1
                check if arr[j] > arr[j+1]
                    set temp = arr[j]
                    set arr[j] = arr[j=1]
                    set arr[j+1] = temp
                end of if
            end of for
        end of for
step 7: for each i = 0 to n
            print the array elements after sorting,arr[i]
step 8: stop
```

Program

```
#include<stdio.h>
void main()
{
   int arr[20],n,i,j,temp;
   printf("Enter the size of the array:");
   scanf("%d",&n);
   printf("Enter the array elements:");
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("Array before sorting:\n");
   for(i=0;i<n;i++)
   {
      printf("%d\t",arr[i]);
   }
   for(i=0;i<n-1;i++)
   {
      for(int j=0;j<n-i-1;j++)
      {
         if(arr[j]>arr[j+1])
         {
            temp=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=temp;
         }
```

```
        }
    }
    printf("\nArray after sorting:\n");
    for(int i=0;i<n;i++)
    {
        printf("%d\t",arr[i]);
    }
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc array_sort.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a.out
Enter the size of the array:5
Enter the array elements:3
1
9
5
7
Array before sorting:
3       1       9       5       7
Array after sorting:
1       3       5       7       9
```

Experiment 7 **Date:17-09-2025**

Searching

Aim:

Program to implement linear search and binary search

<u>Algorithm:</u>

<u>main()</u>

```
step 1:start
step 2: initialize arr[20],n,i,j,item.choice;
step 3: read the array size and array elements,n,arr[i]
step 4: repeat until while choice!=3
    Display:
    1.Linear search
    2.Binar search
    3.Exit

    read users choice
    check if (choice < 3)
        read the element to be searched
    end if

    using switch to perform based on the value of choice
    case 1: call LINEAR(arr,n,item)
            break;
    case 2: call BINARY(arr,n,item)
            break;
    case3: exit(0)
    default: Invalid
```

```
step 5: stop
```

<u>void LINEAR(int arr[],int n,int item)</u>

```
step 1: Start
step 2: input i=0,f=0;
step 3: repeat for i < n
            check if ( item == arr[i])
                  f=1
                  break
            else
                  i=i+1
      end loop
step 4: check if f == 1
            print "Element found at position i"
      else
            print "Element not found"
step 5: stop
```

<u>void BINARY(int arr[],int n,int item)</u>

```
step1: start
step 2: input pos=-1,lb=0,ub=n-1,mid,I,j,temp;
step 3: for each i=0 to n-1
            for each j=0 to n-i-1
                  check if arr[j] > arr[j+1]
                        set temp=arr[j]
                        set arr[j]=arr[j+1]

                        set arr[j+1]=temp

step 4: for each i = 0 to n

            print "the array after sorting",arr[i]

step 5: repeat for lb <= ub

            set mid=(lb+ub)/2

      check if arr[mid] == item

            set pos = mid

            print pos

            break

      else if check arr[mid] > item
            set ub = mid-1

      else

            set lb = mid + 1

step 6: check if pos == -1

            print "element not found"

       else

            print "element found at position pos"

step 7: stop
```

Program

```
#include<stdio.h>

#include<stdlib.h>

void LINEAR(int arr[],int n,int item);

void BINARY(int arr[],int n,int item);

void main()
```

```
{
        int arr[20],n,i,j,item,choice;
        printf("Enter the size of the array:");
        scanf("%d",&n);
        printf("Enter the elements of the array:\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }
        do
        {
                printf("1.Linear Search\n");
                printf("\n2.Binary Search\n");
                printf("\n3.Exit\n");
                printf("\nEnter your choice\n");
                scanf("%d",&choice);
                if (choice < 3)
                {
                        printf("\nEnter the element to be searched: ");
                        scanf("%d",&item);
                }
                switch(choice)
                {
                        case 1:
                                LINEAR(arr,n,item);
                                break;
                        case 2:
                                BINARY(arr,n,item);
                                break;
                        case 3:
                                exit(0);
                        default:
                                printf("Invalid\n");
                }
        }while(choice!=3);
}
void LINEAR(int arr[],int n,int item)
{
        int i=0,f=0;
        while(i<n)
        {
                if(item==arr[i])
                {
                        f=1;
                        break;
                }
                else
                {
                        i=i+1;
                }
        }
        if(f==1)
                printf("Element found at position %d\n",i);
        else
                printf("Element not found\n");
}

void BINARY(int arr[],int n,int item)
```

```
{
	int pos=-1,lb=0,ub=n-1,mid,i,j,temp;
	for(i=0;i<n-1;i++)
	{
		for(j=0;j<n-i-1;j++)
		{
			if(arr[j]>arr[j+1])
			{
				temp=arr[j];
				arr[j]=arr[j+1];
				arr[j+1]=temp;
			}
		}
	}

	printf("Array after sorting: ");
	for(i=0;i<n;i++)
	printf("%d ",arr[i]);
	printf("\n");

	while(lb<=ub)
	{
		mid=(lb+ub)/2;
		if(arr[mid]==item)
		{
			pos=mid;
			break;
		}
		else if(arr[mid]>item)
```

```
                {
                        ub=mid-1;
                }
                else
                {
                        lb=mid+1;
                }
        }

I       f(pos==-1)
        {
                printf("Element not found\n");
        }
        else
        {
                printf("Element found at position %d\n",pos);
        }
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc lin_bin.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a.out
Enter the size of the array:5
Enter the elements of the array:
3
5
9
2
```

```
4
1.Linear Search

2.Binary Search

3.Exit

Enter your choice
1

Enter the element to be searched: 9
Element found at position 2
1.Linear Search

2.Binary Search

3.Exit

Enter your choice
1

Enter the element to be searched: 1
Element not found
1.Linear Search

2.Binary Search

3.Exit
```

```
Enter your choice
2

Enter the element to be searched: 5
Array after sorting: 2 3 4 5 9
Element found at position 3
1.Linear Search

2.Binary Search

3.Exit

Enter your choice
2

Enter the element to be searched: 8
Array after sorting: 2 3 4 5 9
Element not found
1.Linear Search

2.Binary Search

3.Exit

Enter your choice
3
mits@mits-Veriton-M200-H510:~/Faseeh/DS$
```

Experiment 8 **Date:22-09-2025**

Matrix Operations

Aim:

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

Algorithm:

```
step 1: Start
step 2: Input dimensions of Matrix A → r1, c1
step 3: Input dimensions of Matrix B → r2, c2
step 4: Read elements of both matrices
step 5: Display matrices A and B
step 6: Display Menu:
        1. Add
        2. Subtract
        3. Multiply
        4. Exit
step 7:  Repeat until user chooses Exit:
        If choice = 1:
        If r1 == r2 and c1 == c2
        → Add corresponding elements: R[i][j] = A[i][j] + B[i][j]
        Else print "Addition not possible".
        If choice = 2:

        If r1 == r2 and c1 == c2
        → Subtract corresponding elements: R[i][j] = A[i][j] - B[i][j]
        Else print "Subtraction not possible".
        If choice = 3:
```

```
        If c1 == r2
         → Perform matrix multiplication using:
        R[i][j] = Σ (A[i][k] * B[k][j]) for k = 0 to c1 - 1
        Else print "Multiplication not possible".
        If choice = 4: Exit
        Else: print "Invalid choice"
step 8: Stop
```

Program

```
#include <stdio.h>
#include <stdlib.h>
void addMatrices(int r, int c, int A[r][c], int B[r][c], int R[r][c]);
void subMatrices(int r, int c, int A[r][c], int B[r][c], int R[r][c]);
void mulMatrices(int r1, int c1, int A[r1][c1],int r2, int c2, int B[r2][c2],int R[r1][c2]);
void displayMatrix(int r, int c, int M[r][c]);
void main()
{
        int r1, c1;
        int r2, c2;
        int choice;
        printf("FIRST MATRIX IS:\n");
        printf("Enter number of rows for Matrix A: ");
        scanf("%d", &r1);
        printf("Enter number of columns for Matrix A: ");
        scanf("%d", &c1);
        printf("\nSECOND MATRIX IS:\n");
        printf("Enter number of rows for Matrix B: ");
        scanf("%d", &r2);
        printf("Enter number of columns for Matrix B: ");
```

```
scanf("%d", &c2);
int A[r1][c1], B[r2][c2];
printf("\nEnter elements of Matrix A (%d x %d):\n", r1, c1);
for (int i = 0; i < r1; i++)
{
        for (int j = 0; j < c1; j++)
        {
                printf("A[%d][%d]: ", i + 1, j + 1);
                scanf("%d", &A[i][j]);
        }
}
printf("\nEnter elements of Matrix B (%d x %d):\n", r2, c2);
for (int i = 0; i < r2; i++)
{
         for (int j = 0; j < c2; j++)
        {
                printf("B[%d][%d]: ", i + 1, j + 1);
                scanf("%d", &B[i][j]);
        }
}
displayMatrix(r1, c1, A);
displayMatrix(r2, c2, B);

do
{
        printf("1. Add Matrices (A + B)\n");
        printf("2. Subtract Matrices (A - B)\n");
        printf("3. Multiply Matrices (A x B)\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
         switch (choice)
        {
                case 1:
                        if (r1 == r2 && c1 == c2)
                        {
                                int sum[r1][c1];
                                addMatrices(r1, c1, A, B, sum);
                                printf("\nAddition Result (A + B):\n");
                                displayMatrix(r1, c1, sum);
                        }
                        else
                        {
                                printf("ADDITION NOT POSSIBLE\n");
                        }
                        break;
                case 2:
                        if (r1 == r2 && c1 == c2)
                        {
                                int diff[r1][c1];
                                subMatrices(r1, c1, A, B, diff);
                                printf("\nSubtraction Result (A - B) :\n");
                                displayMatrix(r1, c1, diff);
                        }
                        else
                        {
                                printf("\n SUBTRACTION NOT POSSIBLE\n");
                        }
```

```
                        break;
                case 3: {
                         if (c1 == r2)
                        {
                                int product[r1][c2];
                                mulMatrices(r1, c1, A, r2, c2, B, product);
                                printf("\nMultiplication Result (A x B):\n");
                                displayMatrix(r1, c2, product);
                        }
                        else
                        {
                                printf("\n    MULTIPLICATION    NOT    POSSIBLE
%d%d\n", c1, r2);
                        }
                        break;
                }
                case 4:
                        exit(0);
                default:
                        printf("\nInvalid choice.");
        }
    } while (choice != 4);
}
void addMatrices(int r, int c, int A[r][c], int B[r][c], int R[r][c])
{
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            R[i][j] = A[i][j] + B[i][j];
        }
    }
}
void subMatrices(int r, int c, int A[r][c], int B[r][c], int R[r][c])
{
    for (int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            R[i][j] = A[i][j] - B[i][j];
        }
    }
}
void mulMatrices(int r1, int c1, int A[r1][c1],int r2, int c2, int B[r2][c2],int R[r1][c2])
{
    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c2; j++)
        {
            R[i][j] = 0;
            for (int k = 0; k < c1; k++)
            {
                R[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
void displayMatrix(int r, int c, int M[r][c])
{
        printf("Matrix (%d x %d):\n", r, c);
        for (int i = 0; i < r; i++)
        {
                for (int j = 0; j < c; j++)
                {
                        printf("%d\t", M[i][j]);
                }
                printf("\n");
        }
}
```

Output

```
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ gcc matrix.c
mits@mits-Veriton-M200-H510:~/Faseeh/DS$ ./a.out
FIRST MATRIX IS:
Enter number of rows for Matrix A: 3
Enter number of columns for Matrix A: 3

SECOND MATRIX IS:
Enter number of rows for Matrix B: 3
Enter number of columns for Matrix B: 3

Enter elements of Matrix A (3 x 3):
A[1][1]: 1
A[1][2]: 3
A[1][3]: 5
A[2][1]: 7
A[2][2]: 9
```

```
A[2][3]: 2
A[3][1]: 4
A[3][2]: 6
A[3][3]: 8

Enter elements of Matrix B (3 x 3):
B[1][1]: 9
B[1][2]: 8
B[1][3]: 7
B[2][1]: 6
B[2][2]: 5
B[2][3]: 4
B[3][1]: 3
B[3][2]: 2
B[3][3]: 1
Matrix (3 x 3):
1    3    5
7    9    2
4    6    8
Matrix (3 x 3):
9    8    7
6    5    4
3    2    1
1. Add Matrices (A + B)
2. Subtract Matrices (A - B)
3. Multiply Matrices (A x B)
4. Exit
Enter your choice: 1
```

```
Addition Result (A + B):
Matrix (3 x 3):
10    11    12
13    14    6
7    8    9
1. Add Matrices (A + B)
2. Subtract Matrices (A - B)
3. Multiply Matrices (A x B)
4. Exit
Enter your choice: 2

Subtraction Result (A - B) :
Matrix (3 x 3):
-8    -5    -2
1    4    -2
1    4    7
1. Add Matrices (A + B)
2. Subtract Matrices (A - B)
3. Multiply Matrices (A x B)
4. Exit
Enter your choice: 3

Multiplication Result (A x B):
Matrix (3 x 3):
42    33    24
123    105    87
96    78    60
1. Add Matrices (A + B)
2. Subtract Matrices (A - B)
```

```
3. Multiply Matrices (A x B)
4. Exit
Enter your choice: 4
mits@mits-Veriton-M200-H510:~/Faseeh/DS$
```