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

% time	cumulative seconds	self seconds	calls	self Ts/call	total Ts/call	name
0.00	0.00	0.00	1	0.00	0.00	sum

Experiment 4 **Date:29/09/2023**

Types of functions

Aim:

Write a C program to find the sum of two numbers using different types of Functions.

Algorithm:

main()

1.Start

2. Declare c,x,y,z,ch.

3.until

4. Display choices.

5.input c.

a. if c==1: input x,y.

call sum1().

Print z

b. if c==2: input x,y.

call sum2().

c. if c==3:z=call sum3()

print z.

d. if c==3:call sum4().

6.input ch

7.repeat(ch!=0)

8.stop

int sum1(int a,int b)

1.Delcare s

2.set s=a+b

3.return s

4.exit

void sum2(int a,int b)

1.Delcare s

2.set s=a+b

3.print s

4.exit

int sum3()

1.Delcare s,x,y

2.input x,y

3.set s=x+y

4.return s

5.exit

void sum4()

1.Declare x,y,s

2.input x,y

3.set s=x+y

4.print s

5.exit

Program

```
#include<stdio.h>
int sum1(int a,int b)
{
   int s=a+b;
   return s;
}
void sum2(int a,int b)
{
   int s=a+b;
   printf("Sum of numbers: %d",s);
}
int sum3()
{
   int s,x,y;
   printf("Enter two numbers: ");
   scanf("%d%d",&x,&y);
```

```
    s=x+y;
    return s;

}
void sum4()
{
    int s,x,y;
    printf("Enter two numbers: ");
    scanf("%d%d",&x,&y);
    s=x+y;
    printf("Sum of numbers: %d",s);

}
void main()
{
    int c,ch,x,y,z;
    do{
printf("1.Function with argument and return type\n2. Function with
argument but no return type\n3.Function with return type but no
argument\n4. Function with no return type and no argument\nEnter your
choice:");
    scanf("%d", &c);
    switch(c)
    {
        case 1:printf("Enter two numbers: ");
            scanf("%d%d",&x,&y);
            z=sum1(x,y);
            printf("Sum of numbers: %d",z);
            break;
        case 2:printf("Enter two numbers: ");
            scanf("%d%d",&x,&y);
            sum2(x,y);
            break;
        case 3:z=sum3();
            printf("Sum of numbers: %d",z);
            break;
        case 4:sum4();
            break;
    }
    printf("Want to execute more 1-yes/0-no\n");
    scanf("%d",&ch);
    }while(ch!=0);
```

```
}
```

Output

```
1.Function with argument and return type
2. Function with argument but no return type
3.Function with return type but no argument
4. Function with no return type and no
argument
Enter your choice:
1
Enter two numbers:
13 6
Sum of numbers: 19
Want to execute more 1-yes/0-no
1
1.Function with argument and return type
2. Function with argument but no return type
3.Function with return type but no argument
4. Function with no return type and no
argument
Enter your choice:
2
Enter two numbers:
5 7
Sum of numbers: 12
Want to execute more 1-yes/0-no
1
1.Function with argument and return type
2. Function with argument but no return type
3.Function with return type but no argument
4. Function with no return type and no
argument
Enter your choice:
3
Enter two numbers:
6 7
Sum of numbers: 13
Want to execute more 1-yes/0-no
1
1.Function with argument and return type
2. Function with argument but no return type
3.Function with return type but no argument
4. Function with no return type and no
argument
```

```
Enter your choice:
4
Enter two numbers:
3 11
Sum of numbers: 14
Want to execute more 1-yes/0-no 0
```

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:

main()

1. Start
2. Declare a[100],n,c,ch
3. Input n
4. for i=0 to n do

   {

   Input a[i]

   }
5. Display 1.insertion2.deletion 3.traverse
6. Read option into c

a. If c==1 then

call insert(a,n)

b. If c==2 then

call del(a,n)

c. If c==3 then

call traverse(a,n)

7.input ch

8.Repeat 5,6 while ch not equal to 0

void insert(int a[100],int n)

1. Start
2. Declare i,item,k

3. Input item,k
4. if(k>=100)then'
   print array overflow
   else then
   for i=n-1 to k do
   {
   set a[i+1]=a[i]
   set i=i-1
   }
   set a[k]=item
   set n=n+1
5.exit

void del(int a[100],int n)

1. Start
2. Declare j,item,k
3. Input k,item
4. set item=a[k]
5. for j=k to n-1 do
   {
   Set a[j]=a[j+1]
   Set j=j+1
   }
6. print item
7. exit

void traverse(int a[100],int n)

1. Start
2. declare i
3. for i=0 to n do
   {
   Print a[i]
   Set i=i+1
   }
4. exit

Program

```
#include<stdio.h>
void insert(int a[100],int n)
{
```

```
        int i,j,k,item;
        printf("Enter element to be inserted: ");
        scanf("%d",&item);
        printf("Enter location where need to insert: ");
        scanf("%d",&k);
        if(k>=100)
                printf("cannot insert overflow\n");
        else
        {
                for(i=n-1;i>=k;i--)
                a[i+1] = a[i];
                a[k] =item;
                n=n+1;
        }
}
void del(int a[100],int n)
{
        int item,i,j,k;
        printf("Enter location where need to delete:");
        scanf("%d",&k);
        item=a[k];
        for(j=k;j<n-1;j++)
        a[j]=a[j+1];
        n=n-1;
        printf("Deleted element:%d",item);
}
void traverse(int a[100],int n)
{
        int  i;
        printf("Array: \n");
        for(i=0;i<n;++i)
        printf("%d ",a[i]);
}
void main()
{
        int a[100],n,i,k,c,ch,item,j;
        printf("Enter size of array: ");
        scanf("%d",&n);
        printf("Enter elements \n");
        for(i=0;i<n;++i)
        scanf("%d",&a[i]);
        do{
        printf("1.Insert element to a location in array\n2.Delete an element from a
        particular location in array\n3.Traverse an array\nEnter choice\n");
        scanf("%d",&c);
        switch(c)
        {
                case 1:insert(a,n);
```

```
                    break;
                case 2:del(a,n);
                    break;
                case 3:traverse(a,n);
                    break;
                default:printf("Choice is invalid\n");

        }
        printf("Do you want to execute more yes-1/no-0 ");
        scanf("%d",&ch);
        }while(ch!=0);
}
```

Output

```
Enter size of array: 5
Enter elements:
10 20 30 40 50

1.Insert element to a location in array
2.Delete an element from a particular location in array
3.Traverse an array
Enter choice: 1
Enter element to be inserted: 25
Enter location where need to insert: 3
Do you want to execute more yes-1/no-0 1

1.Insert element to a location in array
2.Delete an element from a particular location in array
3.Traverse an array
Enter choice 2
Enter location where need to delete: 4
Deleted element:40
Do you want to execute more yes-1/no-0 1

1.Insert element to a location in array
2.Delete an element from a particular location in array
3.Traverse an array
Enter choice 3
Array: 10 20 25 30 50
Do you want to execute more yes-1/no-0 0
```

Experiment 6 **Date:06/10/2023**

Array Sorting

Aim:

Program to sort an integer array

Algorithm:

main()

1. Start
2. Declare a[100],n,i
3. Input n
4. for i=0 to n do
   {
   input a[i]
   set i=i+1
   }

5.call bubblesort(a,n)

6.for i=0 to n do
{
Print a[i]
Set i=i+1
}

7.stop

void bublesort(int a[],int n)

1.Start
2.Declare temp
3.for i=0 to n-1 do
{
for j=0 to n-i-1 do
{
if(a[j]>a[j+1])then
{
Set temp=a[j]
Set a[j]=a[j+1]
Set a[j+1]=temp
}
}
}
4.exit

Program

```
#include <stdio.h>
void bublesort(int a[],int n)
{
        for(int i=0;i<n-1;++i)
           {
              for(int j=0;j<n-i-1;++j)
              {
                 if(a[j]>a[j+1])
                 {
                    int temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                 }
              }
           }
}
int main()
{
           int a[100],n,i,j,temp;
           printf("Enter limit");
           scanf("%d",&n);
           printf("Enter element\n");
           for(i=0;i<n;++i)
           {
              scanf("%d",&a[i]);
           }
           printf("Unsorted array: ");
           for(i=0;i<n;++i)
           {
              printf("%d ",a[i]);
           }
           bublesort(a,n);
           printf("\nSorted array: ");

          for(i=0;i<n;++i)
           printf("%d ",a[i]);
           return 0;
}
```

Output

Enter limit: 4
Enter element:
3 44 111 0
Unsorted array: 3 44 111 0
Sorted array: 0 3 44 1

Experiment 7 **Date:06/10/2023**

Array Searching

Aim:

To implement linear search and binary search

Algorithm:

main()

1. Start
2. Declare a[100],n,i,s,choice
3. Input n,s
4. for i=0 to n do
   {
   input a[i]
   set i=i+1
   }
5. Display 1.Linear search 2.Binary search 3.Exit
6. Read option into choice
   a.If choice==1 then
   call linearSearch(a,n,s)
   b.If choice==2 then
   call bublesort(a,n)
   call binarySearch(a,n,s)
7. Repeat 5,6 while ch not equal to 3
8. stop

void bublesort(int a[],int n)

1. Start
2. Declare temp
3. for i=0 to n-1 do
   {
   for j=0 to n-i-1 do
   {
   if(a[j]>a[j+1])then
   {
   Set temp=a[j]
   Set a[j]=a[j+1]
   Set a[j+1]=temp
   }
   }
   }
4. exit

void linearSearch(int a[], int n, int s)

1. Start
2. Declare and initialize i,f=0
3. for i=0 to n do
   {
      if (a[i] == s) then
      {
         Set f = 1
         Print i
      }

   }
4. if (f == 0) then
      Print 'Element not found'
5. exit

void binarySearch(int a[], int n, int s)

1. start
2. Declare and initialize l = 0, u = n - 1, pos = -1, mid
3. while(l<=u) do
   {
      Set mid = (l + u) / 2;
      if (s == a[mid]) then
      {
            Set pos = mid
            break
      }
      else if (a[mid] > s)
            set u = mid – 1
      else
            set l = mid + 1

   }
4. if (pos == -1) then
      Print 'Element not found'
   else then
      Print pos
5. exit

Program

```
#include <stdio.h>
void bublesort(int a[],int n)
{
 for(int i=0;i<n-1;++i)
 {
        for(int j=0;j<n-i-1;++j)
        {
                if(a[j]>a[j+1])
                {
                        int temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                }
        }
  }
}
void linearSearch(int a[], int n, int s)
{
 int i, f = 0;
 for (i = 0; i < n; ++i)
 {
        if (a[i] == s)
        {
                f = 1;
                printf("Element present on index: %d\n", i);
                break;
        }
 }
 if (f == 0)
   printf("Element not found\n");
}

void binarySearch(int a[], int n, int s)
{
 int l = 0, u = n - 1, pos = -1, mid;
  while (l <= u)
  {
        mid = (l + u) / 2;
        if (s == a[mid])
        {
                pos = mid;
                break;
        }
        else if (a[mid] > s)
                u = mid – 1;
```

```
        else
                l = mid + 1;
        }
        if (pos == -1)
         printf("Element not found\n");
        else
         printf("Element on index: %d\n", pos);
}

int main()
{
            int a[100], n, choice, s;
            printf("Enter limit: ");
            scanf("%d", &n);
            printf("Enter elements:\n");
            for (int i = 0; i < n; ++i)
               scanf("%d", &a[i]);
            while (choice!=3)
            {
               printf("\nMenu:\n");
               printf("1. Linear Search\n");
               printf("2. Binary Search\n");
               printf("3. Exit\n");
               printf("Enter your choice: ");
               scanf("%d", &choice);
               switch (choice)
               {
                  case 1:printf("Enter element to search: ");
                         scanf("%d", &s);
                         linearSearch(a, n, s);
                         break;
                 case 2: bublesort(a,n);
                         printf("Enter element to search: ");
                         scanf("%d", &s);
                         binarySearch(a, n, s);
                         break;
                  default: printf("Invalid choice, please try again.\n");
                }
            }
            return 0;
        }
```

Output

```
Enter limit :5
Enter elements:
4 2 7 1 6
Menu:
1. Linear Search
2. Binary Search
3. Exit
Enter your choice: 1
Enter element to search: 7
Element present on index: 2
Menu:
1. Linear Search
2. Binary Search
3. Exit
Enter your choice: 2
Enter element to search: 1
Element on index: 0
Menu:
1. Linear Search
2. Binary Search
3. Exit
Enter your choice: 1
Enter element to search: 8
Element not found
```

Experiment 8 **Date:06/10/2023**

Matrix operations

Aim:

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

Algorithm:

1. Start
2. Declare a[5][5],b[5][5],c[10][10],r,m,n,p,q,i,j,k,ch,c
3. input m,n,p,q
4. for i=0 to m do
   {
    for j=0 to n do
    {
        input a[i][j]
    }
   }
5. for i=0 to p do
   {
    for j=0 to q do
    {
        input b[i][j]
    }
   }
6. Display 1.Addition 2.Subtraction 3.Multiplication
7. input ch
   a. if ch==1 then
    {
        If((m==p)&&(n==q))then
        {
            for i=0 to m do
            {
              for j=0 to n do
              {
                  set c[i][j]=a[i][j]+b[i][j]
                  print c[i][j]
              }
            }
        }
    }
   else then
        print ‘invalid’

b. if ch==2 then
{
If((m==p)&&(n==q))then
{
for i=0 to m do
{
for j=0 to n do
{
set c[i][j]=a[i][j]-b[i][j]
print c[i][j]
}
}
}
else then
print 'invalid'
}
c. if ch==3 then
{
if(n==p)then
{
for i=0 to n do
{
for j =0 to q do
{
Set c[i][j]=0
for k=0 to n do
set c[i][j]+=a[i][k]*b[k][j]
print c[i][j]
}
}
}
else then
print 'inavlid'
}

8. input c

9. Repeat step 6,7,8 till c not equal to 0

10. stop

Program

```
#include <stdio.h>
void main()
{
  int a[5][5],b[5][5],c[10][10],m,n,p,q,i,j,ch,r;
```

```
printf("enter size of matrix1\n");
scanf("%d%d",&m,&n);
printf("enter size of matrix2\n");
scanf("%d%d",&p,&q);
printf("enter elements of matrix1");
for(i=0;i<m;++i)
{
   for(j=0;j<n;++j)
      scanf("%d",&a[i][j]);
}
printf("enter elements of matrix2");
for(i=0;i<p;++i)
{
   for(j=0;j<q;++j)
      scanf("%d",&b[i][j]);
}
do{
printf("1.addition\n2.subtraction\n3.multiplication\nenter choice\n");
scanf("%d",&ch);
switch(ch)
{
   case 1:if((m==p)&&(n==q))
          {
            for(i=0;i<m;++i)
            {
               for(j=0;j<n;++j)
               {
                  c[i][j]=a[i][j]+b[i][j];
                  printf("%d ",c[i][j]);
               }
               printf("\n");
            }
         }
         else
          printf("cannot add");
         break;
   case 2:if((m==p)&&(n==q))
         {
            for(i=0;i<m;++i)
            {
               for(j=0;j<n;++j)
               {
                  c[i][j]=a[i][j]-b[i][j];
                  printf("%d ",c[i][j]);
               }
               printf("\n");
            }
          }
```

```
            else
              printf("cannot add");
            break;
      case 3: if(n==p)
            {
            for (i = 0; i < m; ++i)
            {
            for (j = 0; j < q; ++j)
            {
               c[i][j] = 0;
               for (int k = 0; k < n; ++k)
                  c[i][j] += a[i][k] * b[k][j];
               printf("%d ", c[i][j]);
            }
            printf("\n");
            }
            }
            else
            printf("cannot multilply");
           break;
     default:printf("inavlid");
   }
   printf("execute more\n");
   scanf("%d",&r);
   }while(r!=0);
}
```

Output

```
enter size of matrix1
2 2
enter size of matrix2
2 2
enter elements of matrix1
1 2 3 4
enter elements of matrix2
5 6 7 8
1.addition
2.subtraction
3.multiplication
enter choice
1
6 8
10 12
execute more 1
1.addition
2.subtraction
3.multiplication
```

```
enter choice
2

-4 -4
-4 -4
execute more 1
1.addition
2.subtraction
3.multiplication
enter choice
3
19 22
43 50
execute more 0
```

Experiment 9 **Date:12/10/2023**

Stack operations

Aim:

Program to implement stack operations using arrays.

Algorithm:

1. Start
2. Declare and initialize a[100], n,item,t,top=-1,ch,c
3. input n
4. Display 1.Push 2.Pop 3.Display
5. input ch

a. if ch==1 then

```
{
    if(top==n-1)then
        print 'stack overflow'
    else then
    {
        Input item
        Set top=top+1
        Set a[top]=item
    }
}
```

b. if ch==2 then

```
{
    if(top<0)then
        print 'stack underflow'
    else then
    {
        Set item=a[top]
        Set top=top-1
        Print item
    }
}
```

c. if ch==3 then

```
{
    Set t=top
    While(t>=0)do
    {
        Print a[t]
        Set t=t-1
    }
}
```

6. input c
7. repeat 4,5,6 till c not equal to 0
8. stop

Program

```
#include<stdio.h>
void main()
{
   int a[100],n,item,t,top=-1,ch,c;
   printf("enter n");
            scanf("%d",&n);
   do{
   printf("1.push\n2.pop\n3.display\nenter choice\n");
   scanf("%d",&ch);
   switch(ch){
      case 1: if(top==n-1)
               printf("stack overflow\n");
             else{
                printf("enter element to insert\n");
                scanf("%d",&item);
                top=top+1;
                a[top]=item;
                printf("element inserted\n");
             }
              break;
      case 2:if(top<0)
             printf("stack underflow\n");
            else
            {
               item=a[top];
               top=top-1;
               printf("deleted item %d\n",item);
            }
            break;
      case 3:if(top<0)
             printf("stack underflow\n");
            else
            {
                  t=top;
                  while(t>=0)
                  {
                  printf("% d",a[t]);
                  t=t-1;
                  }
            }
            break;
```

```
    default:printf("input is not available\n");

   }
   printf("\nDo you want to execute more yes-1/no-0");
   scanf("%d",&c);
   }while(c!=0);
}
```

Output

```
enter n
3
1.push
2.pop
3.display
enter choice
1
enter element to insert
10
element inserted
Do you want to execute more yes-1/no-0 1

1.push
2.pop
3.display
enter choice
1
enter element to insert
20
element inserted
Do you want to execute more yes-1/no-0 1

1.push
2.pop
3.display
enter choice
1
enter element to insert
30
element inserted
Do you want to execute more yes-1/no-0 1

1.push
2.pop
3.display
enter choice
1
stack overflow
```

Do you want to execute more yes-1/no-0 1

1.push
2.pop
3.display
enter choice
2
deleted item 30
Do you want to execute more yes-1/no-0 1

1.push
2.pop
3.display
enter choice
3
20 10
Do you want to execute more yes-1/no-0 0

Experiment 10 **Date:12/10/2023**

Queue operations

Aim:

Program to implement queue operations using arrays.

Algorithm:

1. Start
2. Declare and initialize q[100], n,item,r,rear=-1,front=-1,ch,c
3. input n
4. Display 1.Enqueue 2.Dequeue 3.Display
5. input ch

a. if ch==1 then
```
{
    if(rear==n-1)then
        print 'queue overflow'
    else then
    {
        if((rear==-1)&&(front==-1)) then
         set front=rear=0
        else then
         set rear=rear+1
        input item
        set q[rear]=item
    }
 }
```
b. if ch==2 then
```
{
    if((front==-1)&&(rear==-1))then
        print 'stack underflow'
    else then
    {
        Set item=q[front]
        Print item
    }
    if(rear==front) then
    set front=rear=1
    else then
     set front=front+1
}
```
c. if ch==3 then
```
{
    Set r=front
    While(r<=rear)do
```

```
            {
                Print q[r]
                Set r=r+1
            }
        }
```

6. input c
7. repeat 4,5,6 till c not equal to 0
8. stop

Program

```
#include<stdio.h>
void main()
{
    int q[100],n,rear=-1,front=-1,item,c,ch;
    printf("Enter n\n");
    scanf("%d",&n);
    do{
    printf("\n1.Enqueue\n2.Dequeue\n3.Display\nEnter choice\n");
    scanf("%d",&ch);
    switch(ch)
    {
    case 1:if(rear==n-1)
            printf("queue overflow\n");
        else
        {
        if((rear==-1)&&(front==-1))
            front=rear=0;
        else
            rear=rear+1;
        printf("Enter element to insert\n");
        scanf("%d",&item);
        q[rear]=item;
        printf("Element inserted\n");
        }
        break;
    case 2:if((front==-1)&&(rear==-1))
            printf("Queue underflow\n");
```

```
            else
            {
             item=q[front];
             printf("Deleted element:%d",item);
            }
            if(rear==front)
            front=rear=1;
            else
            front=front+1;
            break;
        case 3:printf("\nQueue:");
            if((front==-1)&&(rear==-1))
            printf("Queue underflow\n");
            else
            {
               for(int i=front;i<=rear;++i)
               printf("%d ",q[i]);
            }
            break;
        default:printf("input is not available\n");
        }
        printf("\nDo you want to execute more y-1/n-0");
        scanf("%d",&c);
        }while(c!=0);


}
```

Output

```
Enter n
4

1.Enqueue
2.Dequeue
3.Display
Enter choice
1
Enter element to insert
5
Element inserted
Do you want to execute more y-1/n-0 1
```

```
1.Enqueue
2.Dequeue
3.Display
Enter choice
1
Enter element to insert
7
Element inserted
Do you want to execute more y-1/n-0 1

1.Enqueue
2.Dequeue
3.Display
Enter choice
1
Enter element to insert
17
Element inserted
Do you want to execute more y-1/n-0 1

1.Enqueue
2.Dequeue
3.Display
Enter choice
1
Enter element to insert
33
Element inserted
Do you want to execute more y-1/n-0 1

1.Enqueue
2.Dequeue
3.Display
Enter choice
2
Deleted element:5
Do you want to execute more y-1/n-0 1

1.Enqueue
2.Dequeue
3.Display
Enter choice
3
Queue:7 17 33
Do you want to execute more y-1/n-0 0
```

Experiment 11 **Date:12/10/2023**

Circular queue operations

Aim:

Program to implement circular queue operations using arrays.

Algorithm:

1. Start
2. Declare and initialize cq[100], n,item,r,rear=-1,front=-1,ch,c,count=0
3. input n
4. Display 1.Enqueue 2.Dequeue 3.Display
5. input ch

a. if ch==1 then
```
{
    if(count==n)then
        print 'Circular queue overflow'
    else then
    {
        Input item
        Set rear=(rear+1)%n
        Set cq[rear]=item
        Set count=count+1
    }
}
```
b. if ch==2 then
```
{
    if(count==0)then
        print 'Circular queue underflow'
    else then
    {
        Set item=cq[front]
        Print item
        Set front=(front+1)%n
        Set count=count-1
    }
}
```
c. if ch==3 then
```
{
    if count(==0)then
        print 'Circular queue is empty'
    else then
    {
```

```
                    Set i=front
                    Set j=0

                    While(j<count) do
                    {
                      Print cq[i]
                      Set i=(i+1)%n
                      Set j=j+1
                    }
            }
      }
```

6. input c
7. repeat 4,5,6 till c not equal to 0
8. stop

Program

```
#include<stdio.h>
void main()
{
    int i,cq[100],n,rear=0,front=0,item,count=0,ch,c;
    printf("enter n\n");
    scanf("%d",&n);
    do{
    printf("\n1.Enqueue\n2.Dequeue\n3.Display\nEnter your choice\n");
    scanf("%d",&c);
    switch(c)
    {
    case 1:if(count==n)
            printf("circular queue overflow\n");
        else
        {
            printf("enter element to insert\n");
            scanf("%d",&item);
            cq[rear]=item;
            rear=(rear+1)%n;
            count=count+1;
            printf("element inserted\n");
        }
        break;
```

```
        case 2:if(count==0)
                  printf("Circular queue underflow\n");

               else
                {
                 item=cq[front];
                 printf("Element deleted is:%d",item);
                 front=(front+1)%n;
                 count=count-1;
                }
               break;
         case 3:if (count == 0)
                 printf("Circular queue underflow\n");
                 else {
                 printf("\nElements in the Queue are: ");
                 int i = front;
                 int elementsDisplayed =0;
                 while (elementsDisplayed < count) {
                    printf("%d ", cq[i]);
                    i = (i + 1) % n;
                    elementsDisplayed++;
                 }
                 printf("\n");
              }
                   break;
        default:printf("input is not available\n");
        }
        printf("\ndo you want to execute more 1-yes/0-no\n");
        scanf("%d",&ch);
        }while(ch!=0);
}
```

Output

```
enter n
3

1.Enqueue
2.Dequeue
```

```
3.Display
Enter your choice
1
enter element to insert
7
element inserted
do you want to execute more 1-yes/0-no 1

1.Enqueue
2.Dequeue
3.Display
Enter your choice
1
enter element to insert
5
element inserted
do you want to execute more 1-yes/0-no 1

1.Enqueue
2.Dequeue
3.Display
Enter your choice
1
enter element to insert
2
element inserted
do you want to execute more 1-yes/0-no 1

1.Enqueue
2.Dequeue
3.Display
Enter your choice
3
Elements in the Queue are: 7 5 2
do you want to execute more 1-yes/0-no 1

1.Enqueue
2.Dequeue
3.Display
```

Enter your choice
2
Element deleted is:7

do you want to execute more 1-yes/0-no 1

1.Enqueue
2.Dequeue
3.Display
Enter your choice
3

Elements in the Queue are: 5 2

do you want to execute more 1-yes/0-no 0