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 source.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

source.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/s1MCA/sum1
[Thread debugging using libthread_db enabled]
Using host libthread_db library
"/lib/x86_64-linux-gnu/libthread_db.so.1".

Enter 2 numbers : 30 20
Product : 600 [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
a = 10

Experiment 3 **Date: 29.09.2023**

Familiarisation with gprof

Aim:

3. Write a program for finding the sum of two numbers using 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 : 50 20
Sum : 70
mits@mits:~/Desktop/S1MCA/ADS_lab$ gcc -o sum.out -pg sum.c
mits@mits:~/Desktop/S1MCA/ADS_lab$ ./sum.out
Enter 2 numbers : 50 20
Sum : 70
mits@mits:~/Desktop/S1MCA/ADS_lab$ gprof ./sum.out gmon.out >
pgm3.txt

mits@mits:~/Desktop/S1MCA/ADS_lab$ gprof ./sum.out gmon.out > pgm3.txt
Flat profile:
Each sample counts as 0.01 seconds.
```

```
 %   cumulative   self              self     total  time   seconds   seconds    calls
ms/call  ms/call  name
 50.00      0.05     0.05                      main
 25.00      0.07     0.02                      sum
[... More profiling data ...]
Call graph:
index % time    self  children    called     name
                0.00    0.05       1/1       main [3] [... Call graph details ...]
Index by function name:
[... Index details ...]
```

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

Types of functions

Aim:

4. 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("Do you 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
Do you 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 8
Sum of numbers: 13
Do you 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 6
Sum of numbers: 12
Do you 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 12
Sum of numbers: 15
Do you 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.insertion 2.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 delet(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 the location to insert an element: ");
	scanf("%d",&k);
	if(k>=100)
		printf("Array is overflow\n");
	else
	{
		for(i=n-1;i>=k;i--)
		a[i+1] = a[i];
		a[k] =item;
		n=n+1;
	}
}
void delet(int a[100],int n)
{
	int item,i,j,k;
	printf("Enter the location to delete an element:");
	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 the elements: ");
	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 your choice: \n");
```

```
        scanf("%d",&c);
        switch(c)
        {
                case 1:insert(a,n);
                        break;
                case 2:delet(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: 4
Enter the elements: 6 7 8 5

1.Insert element to a location in array
2.Delete an element from a particular location in array
3.Traverse an array
Enter your choice: 1
Enter element to be inserted: 20
Enter location where need to insert: 1
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 your choice: 2
Enter location where need to delete: 2
Deleted element: 7
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 your choice: 3
Array: 6 20 8 5
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)
{
          int temp;
          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 main()
{
             int a[100],n,i,j,temp;
             printf("Enter limit");
             scanf("%d",&n);
             printf("Enter the elements: ");
             for(i=0;i<n;++i)
             {
                scanf("%d",&a[i]);
             }
             bublesort(a,n);
             printf("\nSorted array: ");

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

Output

Enter limit: 4
Enter the elements: 3 25 120 8
Sorted array: 3 8 25 120

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 linear(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 binary(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 linear(int a[], int n, int s)
{
 int i, f = 0;
 for (i = 0; i < n; ++i)
 {
        if (a[i] == s)
        {
                f = 1;
                printf("Element is found at index: %d\n", i);
                break;
        }
 }
 if (f == 0)
   printf("Element is not found\n");
}
void binary(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 is not found\n");
        else
         printf("Element is found at index: %d\n", pos);
}
int main()
{
            int a[100], n, choice, s;
            printf("Enter limit: ");
            scanf("%d", &n);
            printf("Enter the elements: ");
            for (int i = 0; i < n; ++i)
               scanf("%d", &a[i]);
            while (choice!=3)
            {
               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);
                         linear(a, n, s);
                         break;
                 case 2: bublesort(a,n);
                         printf("Enter element to search: ");
                         scanf("%d", &s);
                         binary(a, n, s);
                         break;
                 default: printf("Invalid choice\n");
                }
            }
            return 0;
        }
```

Output

Enter limit :5
Enter the elements: 4 7 3 2 1

1. Linear Search
2. Binary Search
3. Exit
Enter your choice: 1
Enter element to search: 7
Element is found at index: 1
1. Linear Search
2. Binary Search
3. Exit
Enter your choice: 2
Enter element to search: 3
Element is found at index: 2
1. Linear Search
2. Binary Search
3. Exit
Enter your choice: 1
Enter element to search: 8
Element is not found
Enter your choice: 3

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 the size of matrix1: ");
scanf("%d%d",&m,&n);
printf("Enter the size of matrix2: ");
scanf("%d%d",&p,&q);
printf("Enter the elements of matrix1: ");
for(i=0;i<m;++i)
{
   for(j=0;j<n;++j)
      scanf("%d",&a[i][j]);
}
printf("Enter the 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 your 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 Choice");
   }
   printf("Execute more 1-Yes/0-No: ");
   scanf("%d",&r);
   }while(r!=0);
}
```

Output

```
Enter the size of matrix1: 2 2
Enter the size of matrix2: 2 2
Enter the elements of matrix1: 2 3 4 5
Enter the elements of matrix2: 6 7 5 8
1.Addition
2.Subtraction
3.Multiplication
Enter your choice: 1
8 10
9 13
Execute more 1-Yes/0-No: 1
1.Addition
2.Subtraction
3.Multiplication
Enter your choice: 2
```

-4 -4
-1 -3
Execute more 1-Yes/0-No: 1
1.Addition
2.Subtraction
3.Multiplication
Enter your choice: 3
27 38
49 63
Execute more 1-Yes/0-No: 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 the size of stack: ");
            scanf("%d",&n);
   do{
   printf("1.Push\n2.Pop\n3.Display\nEnter your 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("Stack contents are:% d",a[t]);
                  t=t-1;
                  }
            }
            break;
```

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

Output

```
Enter the size of stack:3
1.Push
2.Pop
3.Display
Enter your choice:1
Enter element to insert: 4
Element inserted
Do you want to execute more yes-1/no-0: 1

1.Push
2.Pop
3.Display
Enter your choice:1
Enter element to insert: 9
Element inserted
Do you want to execute more yes-1/no-0: 1

1.Push
2.Pop
3.Display
Enter your choice:1
Enter element to insert: 8
Element inserted
Do you want to execute more yes-1/no-0: 1

1.Push
2.Pop
3.Display
Enter your choice:1
Stack overflow
Do you want to execute more yes-1/no-0: 1

1.Push
2.Pop
3.Display
Enter your choice:2
Deleted item: 8
Do you want to execute more yes-1/no-0: 1
```

1.Push
2.Pop
3.Display
Enter your choice:3
Stack contents are : 9 4
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 'queue 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 the size of queue:\n");
    scanf("%d",&n);
    do{
    printf("\n1.Enqueue\n2.Dequeue\n3.Display\nEnter your 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("Queue contents are: %d ",q[i]);
           }
           break;
      default:printf("Wrong choice\n");
      }
      printf("\nDo you want to execute more y-1/n-0: ");
      scanf("%d",&c);
      }while(c!=0);
}
```

Output

```
Enter the size of queue:3
1.Enqueue
2.Dequeue
3.Display
Enter your choice: 1
Enter element to insert: 8
Element inserted
Do you want to execute more y-1/n-0: 1

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

Enter element to insert: 9
Element inserted
Do you want to execute more y-1/n-0: 1

1.Enqueue
2.Dequeue
3.Display
Enter your 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 your choice: 1
Queue overflow
Do you want to execute more y-1/n-0 1

1.Enqueue
2.Dequeue
3.Display
Enter your choice: 3
Queue contents are: 8 9 7
Do you want to execute more y-1/n-0:1

1.Enqueue
2.Dequeue
3.Display
Enter your choice: 2
Deleted element: 8
Do you want to execute more y-1/n-0: 1

1.Enqueue
2.Dequeue
3.Display
Enter your choice: 3
Queue contents are: 9 7
Do you want to execute more y-1/n-0: 0

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

Circular Queue Operations

Aim:

11. Program to implement circular queue using array.

Algorithm:

main()

1. Start
2. Declare ch,a[50],f=-1,r=-1 and n.
3. Display choices.
4. Read option ch.
    a. if ch==1 call enqueue().
    b. if ch==2 call dequeue().
    c. if ch==3 call display()
5. Repeat steps 3 while ch>0&&ch<4.
6. Stop.

void enqueue()

1. Start
2. if (r+1)%n==f print queue is full
3. else
    if f==-1
        set f=r=0
    else
        r=(r+1)%n;
    read a[r]
4. Exit

void dequeue()

1. Start
2. if f==-1 print queue is empty
3. else
    print a[f] is deleted
    if(f==r)
        set f=r=-1
    else
        f=(f+1)%n
4. Exit

void display()

1. Start
2. if f==-1 print queue underflow
3. else
   for(i=f;i!=r;i=(i+1)%n){
   printf("%d ",a[i]);
   }
   printf("%d ",a[i]);
4. Exit.

Program

```
#include<stdio.h>
#define n 5
int a[50],f=-1,r=-1;
void enqueue(){
if((r+1)%n==f){
printf("Queue is full");
}
else{
if(f==-1)
f=r=0;
else
r=(r+1)%n;
printf("Enter the element to be inserted:");
scanf("%d",&a[r]);
}
}
void dequeue(){
if(f==-1){
printf("Queue is empty");
}
else{
printf("%d is deleted",a[f]);
if(f==r)
f=r=-1;
else
f=(f+1)%n;
}
}
void display(){
```

```
int i;
if(f==-1){
printf("Queue is empty");
}
else{
printf("Queue:");
for(i=f;i!=r;i=(i+1)%n){
printf("%d ",a[i]);
}
printf("%d ",a[i]);
}
}
void main(){
int ch;
do{
printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice:");
scanf("%d",&ch);
switch(ch){
case 1 : enqueue();
break;
case 2 : dequeue();
break;
case 3 : display();
break;
}
}while(ch>0&&ch<4);
}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc cq.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Enter the element to be inserted:2

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
```

```
Enter the element to be inserted:5

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Enter the element to be inserted:8

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:3
Queue:2 5 8

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:2
2 is deleted

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:3
Queue:5 8

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:2
5 is deleted

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:3
Queue:8

1. Insert
2. Delete
3. Display
```

```
4. Exit
Enter your choice:1
Enter the element to be inserted:9

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Enter the element to be inserted:6

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:1
Enter the element to be inserted:4

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:3
Queue:8 9 6 4

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:4
akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 12 **Date: 19/10/2023**

Singly Linked List Operations

Aim:

12. To implement the following operations on a singly linked list
    a. Creation
    b. Insert a new node at front
    c. Insert an element after a particular
    d. Deletion from beginning
    e. Deletion from the end
    f. Searching
    g. Traversal.

Algorithm:

main()

1. Start
2. struct node{
   int data;
   struct node *next;
   }*head, *ptr, *temp;
3. Display choices.
4. Read option ch.
   a. if ch==1 call ins_beg().
   b. if ch==2 call ins_spec().
   c. if ch==3 call del_beg()
   d. if ch==4 call del_end()
   e. if ch==5 call search()
   f. if ch==6 call display()
5. Repeat step 3 while ch>0&&ch<7.
6. Stop.

void ins_beg()

1. Start
2. ptr = malloc(sizeof(struct node))
3. Read ptr->data
   if head==NULL
   ptr->next=NULL;
   head=ptr
   else
   ptr->next=head;

head=ptr

4. Exit

void ins_spec()

1. Start
2. ptr = malloc(sizeof(struct node))
3. Read ptr->data
4. set temp=head
5. for(int i=1;i<p;i++){
   temp=temp->next;
   if(temp==NULL){
   printf("Invalid Position");
   break;
   }
   }
   ptr->next=temp->next;
   temp->next=ptr;
6. Exit

void del_beg()

1. Start
2. if head==NULL print List Empty
3. else
   print head->data is deleted
   if head->next==NULL
   free(head);
   head=NULL;
   else
   ptr=head;
   head=ptr->next;
   free(ptr);
4. Exit.

void del_end()

1. Start
2. if head==NULL print List Empty
3. else
   if head->next==NULL
   print head->data is deleted
   free(head);
   head=NULL;
   else
   ptr=head;

```
                while(ptr->next!=NULL){
                        temp=ptr;
                        ptr=ptr->next;
                }
        printf("%d is deleted",ptr->data);
        temp->next=NULL;
        free(ptr);
```

4. Exit.

void display()

1. Start
2. if head==NULL print List Empty
3. else

```
        printf("Linked List : ");
        while(ptr!=NULL){
                printf("%d ",ptr->data);
                ptr=ptr->next;
        }
```

4. Exit.

void search()

1. Start
2. Declare x,i=1,f=0
3. if head==NULL print List Empty
4. else

```
        read x
        for(ptr=head; ptr!=NULL; ptr=ptr->next){
                if(ptr->data==x){
                        print element found at node i
                        set f=1
                }
                i++
        }
        if f ==0 print Element not found
```

5. Exit.

Program

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node *next;
}*head, *ptr, *temp;
void ins_beg(){
ptr = malloc(sizeof(struct node));
printf("Enter the item : ");
scanf("%d",&ptr->data);
if(head==NULL){
ptr->next=NULL;
head=ptr;
}
else{
ptr->next=head;
head=ptr;
}
}
void ins_spec(){
int p;
ptr = malloc(sizeof(struct node));
printf("Enter the item and it's position : ");
scanf("%d %d",&ptr->data,&p);
temp=head;
for(int i=1;i<p;i++){
temp=temp->next;
if(temp==NULL){
printf("Invalid Position");
break;
}
}
ptr->next=temp->next;
temp->next=ptr;
}
void del_beg(){
if(head==NULL){
printf("List is empty");
}
```

```
else{
printf("%d is deleted",head->data);
if(head->next==NULL){
free(head);
head=NULL;
}
else{
ptr=head;
head=ptr->next;
free(ptr);
}
}
}
void del_end(){
if(head==NULL){
printf("List Empty");
}
else{
if(head->next==NULL){
printf("%d is deleted",head->data);
free(head);
head=NULL;
}
else{
ptr=head;
while(ptr->next!=NULL){
temp=ptr;
ptr=ptr->next;
}
printf("%d is deleted",ptr->data);
temp->next=NULL;
free(ptr);
}
}
}
void display(){
if(head==NULL){
printf("List is empty");
}
else{
ptr=head;
printf("Linked List : ");
```

```
while(ptr!=NULL){
printf("%d ",ptr->data);
ptr=ptr->next;
}
}
}
void search(){
int x,i=1,f=0;
if(head==NULL){
printf("List is empty");
}
else{
printf("Enter the item : ");
scanf("%d",&x);
for(ptr=head; ptr!=NULL; ptr=ptr->next){
if(ptr->data==x){
printf("Element found at node %d",i);
f=1;
}
i++;
}
if(f==0){
printf("Element not found");
}
}
}
void main(){
int ch;
do{
printf("\n1. Insert at front\n2. Insert at Specific Position\n3. Delete at front\n4. Delete
at rear\n5. Search\n6. Display\n7. Exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch){
case 1: ins_beg();
break;
case 2: ins_spec();
break;
case 3: del_beg();
break;
case 4: del_end();
break;
case 5: search();
```

```
break;
case 6: display();
break;
}
}while(ch>0&&ch<7);}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc Singly.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 1
Enter the item : 2

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 1
Enter the item : 3

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 1
Enter the item : 4
```

```
1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 6
Linked List : 4 3 2


1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 2
Enter the item and it's position : 5 1

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 6
Linked List : 4 5 3 2


1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 3
4 is deleted
```

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 6
Linked List : 5 3 2

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 4
2 is deleted

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 6
Linked List : 5 3

1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search

```
6. Display
7. Exit
Enter your choice: 5
Enter the item : 3
Element found at node 2


1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 6
Linked List : 5 3


1. Insert at front
2. Insert at Specific Position
3. Delete at front
4. Delete at rear
5. Search
6. Display
7. Exit
Enter your choice: 7
akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 13 **Date: 20/10/2023**

Doubly Linked List Operations

Aim:

13. To implement the following operations on a singly linked list
    a. Creation
    b. Count the number of nodes
    c. Insert a new node at front
    d. Insert an element at end
    e. Deletion from beginning
    f. Deletion from the end
    g. Searching
    h. Traversal.

Algorithm:

main()

1. Start
2. struct node{
   int data;
   struct node *l, *r;
   }*head, *ptr, *temp;
   c=0
3. Display choices.
4. Read option ch.
   a. if ch==1 call ins_beg().
   b. if ch==2 call ins_end().
   c. if ch==3 call del_beg()
   d. if ch==4 call del_end()
   e. if ch==5 call search()
   f. if ch==6 print c
   g. if ch==7 call display()
5. Repeat step 3 while ch>0&&ch<8.
6. Stop.

void ins_beg()

1. Start
2. ptr = malloc(sizeof(struct node))
3. Read ptr->data
4. c++
   if head==NULL

```
                ptr->r=ptr->l=NULL;
                head=ptr
        else
                ptr->l=NULL;
                ptr->r=head;
                head=ptr;
```

5. Exit

void ins_end()

1. Start
2. ptr = malloc(sizeof(struct node))
3. Read ptr->data
4. c++

```
        if head==NULL
                ptr->r=ptr->l=NULL;
                head=ptr
        else
                temp=head;
                while(temp->r!=NULL){
                        temp=temp->r;
                }
                temp->r=ptr;
                ptr->l=temp;
                ptr->r=NULL;
```

5. Exit

void del_beg()

1. Start
2. if head==NULL print List Empty
3. else

```
        c--
        print head->data is deleted
        if(head->r==NULL){
                free(head);
                head=NULL;
        }
        else{
                ptr=head;
                head=head->r;
                head->l=NULL;
                free(ptr);
        }
```

4. Exit.

void del_end()

1. Start
2. if head==NULL print List Empty
3. else

```
c--;
if(head->r==NULL){
        printf("%d is deleted",head->data);
        free(head);
        head=NULL;
}
else{
        ptr=head;
        while(ptr->r!=NULL){
                ptr=ptr->r;
        }
        printf("%d is deleted",ptr->data);
        ptr->l->r=NULL;
        free(ptr);
}
```

4. Exit.

void display()

1. Start
2. if head==NULL print List Empty
3. else

```
printf("Linked List : ");
while(ptr!=NULL){
        printf("%d\t",ptr->r);
        ptr=ptr->r;
}
```

4. Exit.

void search()

1. Start
2. Declare x,i=1,f=0
3. if head==NULL print List Empty
4. else

```
read x
for(ptr=head; ptr!=NULL; ptr=ptr->r){
        if(ptr->r==x){
                print element found at node i
                set f=1
        }
```

i++
}
if f ==0 print Element not found

5. Exit.

Program

```
#include<stdio.h>
#include<stdlib.h>
int c=0;
struct node{
int data;
struct node *l, *r;
}*head, *ptr, *temp;
void ins_beg(){
ptr = malloc(sizeof(struct node));
printf("Enter the item: ");
scanf("%d",&ptr->data);
c++;
if(head==NULL){
ptr->r=ptr->l=NULL;
head=ptr;
}
else{
ptr->l=NULL;
ptr->r=head;
head=ptr;
}
}
void ins_end(){
ptr = malloc(sizeof(struct node));
printf("Enter the item: ");
scanf("%d",&ptr->data);
c++;
if(head==NULL){
ptr->r=ptr->l=NULL;
head=ptr;
}
else{
temp=head;
while(temp->r!=NULL){
```

```
temp=temp->r;
}
temp->r=ptr;
ptr->l=temp;
ptr->r=NULL;
}
}
void del_beg(){
if(head==NULL){
printf("List is empty");
}
else{
c--;
printf("%d is deleted",head->data);
if(head->r==NULL){
free(head);
head=NULL;
}
else{
ptr=head;
head=head->r;
head->l=NULL;
free(ptr);
}
}
}
void del_end(){
if(head==NULL){
printf("List is empty");
}
else{
c--;
if(head->r==NULL){
printf("%d is deleted",head->data);
free(head);
head=NULL;
printf("\n");
}
else{
ptr=head;
while(ptr->r!=NULL){
ptr=ptr->r;
```

```
}
printf("%d is deleted",ptr->data);
ptr->l->r=NULL;
free(ptr);
printf("\n");
}
}
}
void display(){
ptr=head;
if(ptr==NULL){
printf("List is empty");
printf("\n");
}
else{
printf("Doubly Linked List: ");
while(ptr!=NULL){
printf("%d ",ptr->data);
ptr=ptr->r;
}
}
printf("\n");
}
void search(){
int x,i=1,f=0;
if(head==NULL){
printf("List is empty");
}
else{
printf("Enter the item: ");
scanf("%d",&x);
for(ptr=head; ptr!=NULL; ptr=ptr->r){
if(ptr->data==x){
printf("Element found at node %d",i);
f=1;
}
i++;
}
if(f==0){
printf("Element not found");
}
}
```

```
}
void main(){
int ch;
do{
printf("\n1. Insert at front\n2. Insert at rear\n3. Delete at front\n4. Delete at rear\n5.
Display\n6. Search\n7. Count\n8. Exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch){
case 1: ins_beg();
break;
case 2: ins_end();
break;
case 3: del_beg();
break;
case 4: del_end();
break;
case 5: display();
break;
case 6: search();
break;
case 7: printf("Number of nodes: %d",c);
break;
}
}while(ch>0&&ch<8);

}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc Doubly.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 1
```

```
Enter the item: 2

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 1
Enter the item: 3

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 1
Enter the item: 4

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 2
Enter the item: 9

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
```

```
8. Exit
Enter your choice: 2
Enter the item: 8

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 5
Doubly Linked List: 4 3 2 9 8

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 3
4 is deleted


1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 5
Doubly Linked List: 3 2 9 8

1. Insert at front
2. Insert at rear
3. Delete at front
```

4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 4
8 is deleted

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 5
Doubly Linked List: 3 2 9

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 6
Enter the item: 6
Element not found

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit

```
Enter your choice: 7
Number of nodes: 3

1. Insert at front
2. Insert at rear
3. Delete at front
4. Delete at rear
5. Display
6. Search
7. Count
8. Exit
Enter your choice: 8
akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 14 **Date: 27/10/2023**

Linked Stack Operations

Aim:

14. To implement a menu driven program to perform following stack operations using linked list
    a. Push
    b. Pop
    c. Traversal

Algorithm:

main()

1. Start
2. struct node{
   int data;
   struct node *next;
   }*top, *ptr;
3. Display choices.
4. Read option ch.
   a. if ch==1 call push().
   b. if ch==2 call pop().
   c. if ch==3 call display()
5. Repeat step 3 while ch>0&&ch<4.
6. Stop.

void push()

1. Start
2. ptr = malloc(sizeof(struct node))
3. Read ptr->data
4. ptr->next=top; top=ptr;
5. Exit

void pop()

1. Start
2. if head==NULL print Stack Underflow
3. else
   ptr=top
   print ptr->data is deleted
   top=top->next;
   free(ptr);
4. Exit.

void display()

1. Start
2. if head==NULL print Stack Empty
3. else
   while(ptr!=NULL){
   print ptr->data
   ptr=ptr->next
   }
4. Exit.

Program

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node *next;
}*top, *ptr;
void push(){
ptr = malloc(sizeof(struct node));
printf("Enter the item: ");
scanf("%d",&ptr->data);
ptr->next=top;
top=ptr;
}
void pop(){
if(top==NULL){
printf("Stack Underflow");
}
else{
ptr=top;
printf("%d is deleted",ptr->data);
top=top->next;
free(ptr);
}
}
void display(){
ptr=top;
if(ptr==NULL){
printf("Stack Empty");
}
else{
printf("Stack: ");
```

```
while(ptr!=NULL){
printf("%d ",ptr->data);
ptr=ptr->next;
}
}
printf("\n");
}
void main(){
int ch;
do{
printf("\n1.Push\n2.Pop\n3.Display\n4.Exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch){
case 1: push();
break;
case 2: pop();
break;
case 3: display();
break;
}
}while(ch>0&&ch<4);
}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc LinkedStack.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 1
Enter the item: 6

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 1
Enter the item: 3

1.Push
2.Pop
3.Display
```

```
4.Exit
Enter your choice: 1
Enter the item: 5

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 1
Enter the item: 9

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 3
Stack: 9 5 3 6

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 2
9 is deleted

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 1
Enter the item: 8

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 3
Stack: 8 5 3 6

1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 2
8 is deleted
```

```
1.Push
2.Pop
3.Display
4.Exit
Enter your choice: 4
akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 15 **Date: 27/10/2023**

Linked Queue Operations

Aim:

15. To implement a menu driven program to perform following queue operations using linked list
    a. Enqueue
    b. Dequeue
    c. Traversal

Algorithm:

main()

1. Start
2. struct node{
   int data;
   struct node *next;
   }*top, *ptr, *f, *r;
3. Display choices.
4. Read option ch.
   a. if ch==1 call enqueue().
   b. if ch==2 call dequeue().
   c. if ch==3 call display()
5. Repeat step 3 while ch>0&&ch<4.
6. Stop.

void enqueue()

1. Start
2. ptr = malloc(sizeof(struct node))
3. Read ptr->data
4. if f==NULL
   f=r=ptr;
   else
   r->next=ptr;
   r=ptr;
5. Exit

void dequeue()

1. Start
2. if f==NULL print Queue is empty
3. else
   ptr=f

print ptr->data is deleted
f=ptr->next;
free(ptr);

4. Exit.

void display()

1. Start
2. if head==NULL print Queue is empty
3. else
   while(ptr!=NULL){
   print ptr->data
   ptr=ptr->next
   }
4. Exit.

Program

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node *next;
}*top, *ptr, *f, *r;
void enqueue(){
ptr = malloc(sizeof(struct node));
printf("Enter the item: ");
scanf("%d",&ptr->data);
if(f==NULL){
f=r=ptr;
}
else{
r->next=ptr;
r=ptr;
}
}
void dequeue(){
if(f==NULL){
printf("Queue is empty");
}
else{
ptr=f;
printf("%d is deleted",ptr->data);
f=ptr->next;
```

```
free(ptr);
}
printf("\n");
}
void display(){
if(f==NULL){
printf("Queue is empty");
}
else{
ptr=f;
printf("Queue: ");
while(ptr!=NULL){
printf("%d ",ptr->data);
ptr=ptr->next;
}
}
printf("\n");
}
void main(){
int ch;
do{
printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch){
case 1: enqueue();
break;
case 2: dequeue();
break;
case 3: display();
break;
}
}while(ch>0&&ch<4);
}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc LinkedQueue.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item: 4
```

```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item: 7

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item: 3

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue: 4 7 3

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item: 8

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item: 4

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue: 4 7 3 8 4

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

```
Enter your choice: 2
4 is deleted

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
7 is deleted

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue: 3 8 4

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 16 **Date: 02/11/2023**

Binary Search Tree Operations

Aim:

16. Menu Driven program to implement Binary Search Tree (BST) and to perform following operations
    a. Insertion of a node.
    b. Deletion of a node.
    c. In-order traversal.
    d. Pre-order traversal.
    e. Post-order traversal.

Algorithm:

main()

1. Start
2. struct node{
   int data;
   struct node *l,*r;
   }*root, *ptr, *succ, *succparent;
3. Declare ch and x
4. Display choices.
5. Read option ch.
   a. if ch==1 read x and call root=insert(root,x).
   b. if ch==2 read x and call root=del(root,x).
   c. if ch==3 call inorder(root)
   d. if ch==4 call preorder(root)
   e. if ch==5 call postorder(root)
6. Repeat step 3 while ch>0&&ch<6.
7. Stop.

struct node* create(int x)

1. Start
2. ptr=malloc(sizeof(struct node));
3. ptr->data=x;
4. ptr->l=ptr->r=NULL;
5. return ptr;
6. Exit

struct node* insert(struct node* root, int x)

1. Start
2. if root==NULL return create(x)
3. if x>root->data
   root->r=insert(root->r,x);
   else
   root->l=insert(root->l,x);
4. return root;
5. Exit.

struct node* del(struct node* root, int x)

1. Start
2. if root==NULL return root
3. if x>root->data
   root->r=del(root->r,x)
   return root
   else if x<root->data
   root->l=del(root->l,x)
   return root
4. if root->l==NULL
   ptr=root->r
   free(root)
   return ptr
   else if root->r==NULL
   ptr=root->l
   free(root)
   return ptr
5. succparent=root
   succ=root->r;
   while(succ->l!=NULL){
   succparent=succ;
   succ=root->l;
   }
6. if succparent!=root
   succparent->l=succ->r
   else
   succparent->r=succ->r
7. root->data=succ->data
8. free(succ);
9. return root;
10. Exit.

void inorder(struct node* root)

1. Start
2. if(root!=NULL)
   inorder(root->l)
   print root->data
   inorder(root->r)
3. Exit.

void preorder(struct node* root)

1. Start
2. if(root!=NULL)
   print root->data
   inorder(root->l)
   inorder(root->r)
3. Exit.

void postorder(struct node* root)

1. Start
2. if(root!=NULL)
   inorder(root->l)
   inorder(root->r)
   print root->data
3. Exit.

Program

```
#include<stdio.h>
#include <stdlib.h>
struct node{
int data;
struct node *l,*r;
}*root, *ptr, *succ, *succparent;
struct node* create(int x){
ptr=malloc(sizeof(struct node));
ptr->data=x;
ptr->l=ptr->r=NULL;
return ptr;
}
struct node* insert(struct node* root, int x){
if(root==NULL){
return create(x);
```

```
}
if(x>root->data){
root->r=insert(root->r,x);
}
else{
root->l=insert(root->l,x);
}
return root;
}
struct node* del(struct node* root, int x){
if(root==NULL){
return root;
}
if(x>root->data){
root->r=del(root->r,x);
return root;
}
else if(x<root->data){
root->l=del(root->l,x);
return root;
}
if(root->l==NULL){
ptr=root->r;
free(root);
return ptr;
}
else if(root->r==NULL){
ptr=root->l;
free(root);
return ptr;
}
succparent=root;
succ=root->r;
while(succ->l!=NULL){
succparent=succ;
succ=root->l;
}
if(succparent!=root){
succparent->l=succ->r;
}
else{
succparent->r=succ->r;
```

```
}
root->data=succ->data;
free(succ);
return root;
}
void inorder(struct node* root){
if(root!=NULL){
inorder(root->l);
printf("%d ",root->data);
inorder(root->r);
}
}
void preorder(struct node* root){
if(root!=NULL){
printf("%d ",root->data);
inorder(root->l);
inorder(root->r);
}
}
void postorder(struct node* root){
if(root!=NULL){
inorder(root->l);
inorder(root->r);
printf("%d ",root->data);
}
}
void main(){
int ch,x;
do{
printf("\n1. Insert\n2. Delete\n3. Inorder Traversal\n4. Preorder Traversal\n5.
Postorder Traversal\n6. Exit\nEnter your choice: ");
scanf("%d",&ch);
switch(ch){
case 1: printf("Enter the element: ");
scanf("%d",&x);
root=insert(root,x);
break;
case 2: printf("Enter the element: ");
scanf("%d",&x);
root=del(root,x);
break;
case 3: printf("Inorder Traversal: ");
```

```
inorder(root);
break;
case 4: printf("Preorder Traversal: ");
preorder(root);
break;
case 5: printf("Postorder Traversal: ");
postorder(root);
break;
}
}while(ch>0&&ch<6);

}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc BinarySearch.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 23

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 34

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
```

```
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 45

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 56

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 67

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 78

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 89

1. Insert
```

```
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 3
Inorder Traversal: 23 34 45 56 67 78 89

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 4
Preorder Traversal: 23 34 45 56 67 78 89

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 5
Postorder Traversal: 34 45 56 67 78 89 23

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the element: 12

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
```

```
Enter your choice: 3
Inorder Traversal: 12 23 34 45 56 67 78 89

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 4
Preorder Traversal: 23 12 34 45 56 67 78 89

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 5
Postorder Traversal: 12 34 45 56 67 78 89 23

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 2
Enter the element: 67

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 3
Inorder Traversal: 12 23 34 45 56 78 89

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
```

```
5. Postorder Traversal
6. Exit
Enter your choice: 4
Preorder Traversal: 23 12 34 45 56 78 89

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 5
Postorder Traversal: 12 34 45 56 78 89 23

1. Insert
2. Delete
3. Inorder Traversal
4. Preorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 6
akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 17 **Date: 09/11/2023**

Bitstring Operations

Aim:

17. To implement set operations using bit strings.

Algorithm:

main()

1. Start
2. Declare int a[11], b[11], res[11], U[11]={1,2,3,4,5,6,7,8,9,10},s1,s2,ch;
3. Read size of bit-string 1 s1
4. Call input(a,s1) and display(a)
5. Read size of bit-string 2 s2
6. Call input(b,s2) and display(b)
7. Display choices.
8. Read option ch.
    a. if ch==1 call set_union().
    b. if ch==2 call set_intersection().
    c. if ch==3 call set_difference().
    d. if ch==3 if(set_equality())
        print Bit strings are equal.
    else
        print Bit strings are not equal.
9. Repeat step 3 while ch>0&&ch<4.
10. Stop.

void set_union()

1. Start
2. for(int i=1;i<11;i++)
        res[i]=a[i] | b[i];
3. display(res)
4. Exit.

void set_intersection()

1. Start
2. for(int i=1;i<11;i++)
        res[i]=a[i] & b[i];
3. display(res)
4. Exit.

void set_union()

1. Start
2. for(int i=1;i<11;i++)
   res[i]=a[i] & ~b[i];
3. display(res)
4. Exit.

bool set_equality()

1. Start
2. for(int i=1;i<11;i++)
   if a[i] != b[i]
   return false
3. return true
4. Exit.

void input(int bs[], int n)

1. Start
2. Declare x
3. for(int i=1;i<11;i++)
   read x
   bs[x]=1
4. Exit.

void display(int bs[])

1. Start
2. for(int i=1;i<11;i++)
   print bs[i]
3. Exit.

Program

```
#include<stdio.h>
#include <stdbool.h>
int a[11], b[11], res[11];
int U[11]={1,2,3,4,5,6,7,8,9,10};

void display(int bs[]){
   for(int i=1;i<11;i++){
      printf("%d ",bs[i]);
```

```
    }
 }
void input(int bs[], int n){
    int x;
    printf("Enter the elements: ");
    for(int i=0;i<n;i++){
       scanf("%d",&x);
       bs[x]=1;
    }
 }

 void set_union(){
    for(int i=1;i<11;i++){
       res[i]=a[i] | b[i];
    }

    printf("\nUnion Set: ");
    display(res);
 }

 void set_intersection(){
    for(int i=1;i<11;i++){
       res[i]=a[i] & b[i];
    }

    printf("\nIntersection Set: ");
    display(res);
 }

 void set_difference(){
    for(int i=1;i<11;i++){
       res[i]=a[i] & ~b[i];
    }

    printf("\nDifference Set: ");
    display(res);
 }
          bool set_equality(){
    for(int i=1;i<11;i++){
       if(a[i] != b[i]){
          return false;
       } }
    return true;}
```

Output

```
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ gcc BitString.c
akhila@akhila-VirtualBox:~/Desktop/S1MCA$ ./a.out

Enter the size of bit-string 1: 5
Enter the elements: 1 3 5 7 9
Set A: 1 0 1 0 1 0 1 0 1 0

Enter the size of bit-string 2: 5
Enter the elements: 2 4 6 8 10
Set B: 0 1 0 1 0 1 0 1 0 1

1. Union
2. Intersection
3. Difference
4. Equality
5. Exit
Enter your choice: 1
Union Set : 1 1 1 1 1 1 1 1 1 1

1. Union
2. Intersection
3. Difference
4. Equality
5. Exit
Enter your choice: 2
Intersection Set: 0 0 0 0 0 0 0 0 0 0

1. Union
2. Intersection
3. Difference
4. Equality
5. Exit
Enter your choice: 3
Difference Set: 1 0 1 0 1 0 1 0 1 0

1. Union
2. Intersection
3. Difference
4. Equality
5. Exit
```

```
Enter your choice: 4
Bit strings are not equal

1. Union
2. Intersection
3. Difference
4. Equality
5. Exit
Enter your choice: 5

akhila@akhila-VirtualBox:~/Desktop/S1MCA$
```

Experiment 18 **Date:15/12/2023**

Graph Traversal

Aim:

Write a program to implement BFS and DFS on a connected undirected graph

Algorithm:

main()

1.start
2. Declare variables n, i, s, ch, j, c, dummy,a[20][20],vis[20]
3. input n
4. for i = 1 to n do
    for j = 1 to n do
        input a[i][j]
5.repeat step 6,7,8 till ch not equal to 'n'
6.for i=0 to n do
    vis[i]=0
7.input ch
  a.if(ch==1)then
    {
      Call bfs(s,n)
    }
  b.if(ch==2)then
    {
      Call b=dfs(s,n)
    }
8.input ch
9.stop

void bfs(int s,int n)

1.start
2.declare p,i
3.call enqueue(s)
4.set vis[s]=1
5.p=dequeue()
6.if(p!=0)then
    print p
7.while(p!=0)do
  {
    for i = 1 to n do
    {
      if((a[p][i]!=0)&&(vis[i]==0))then
      {

```
                        call enqueue(i)
                        set vis[i]=1
                    }
            set p=dequeue()
            if(p!=0)then
               print p

      }
    8.for i = 1 to n do
            if(vis[i]==0)then
                    call bfs(i,n)
    9.exit
```

void enqueue(int item)

```
    1.start
    2.if(rear==19)then
            print "QUEUE FULL"
      else then
      {

            if(rear==-1)then
            {
                    set q[++rear]=item
                    set front++
            }
            else then
                    set q[++rear]=item
      }
     3.exit
```

int dequeue()

```
    1.start
    2.declare k
    3.if((front>rear)||(front==-1))then
            return(0)
    else then
    {
            set k=q[front++]
            return(k)
    }
    4.exit
```

void dfs(int s,int n)

```
    1.start
    2.declare i,k
    3.call push(s)
    4.set vis[s]=1
```

```
5.set k=pop()
6.if(k!=0)then
        print k
7.while(k!=0)do
  {
        for i=1 to n do
        {
              if((a[k][i]!=0)&&(vis[i]==0))then
              {
                    call push(i)
                     vis[i]=1
              }
        }
        set k=pop()
        if(k!=0)then
            print k
}
8.for i = 1 to n do
  {
        if(vis[i]==0)then
             call dfs(i,n)
 }
9.exit
```

void push(int item)

```
1.start
2.if(top==19)then
        print "Stack overflow"
  else then
        set stack[++top]=item
3.exit
```

int pop()

```
1.start
2.declare k
3.if(top==-1)then
        return(0)
   else then
   {
        set k=stack[top--]
        return(k)
   }
 4.exit
```

Program

```
#include<stdio.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
```

```
int dequeue();
void enqueue(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();
void main()
{
int n,i,s,ch,j;
char c,dummy;
printf("ENTER THE NUMBER VERTICES ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
scanf("%d",&a[i][j]);
}
}
printf("THE ADJACENCY MATRIX IS\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf(" %d",a[i][j]);
}
printf("\n");
}
do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);
switch(ch)
{
case 1:bfs(s,n);
        break;
case 2:dfs(s,n);
        break;
```

```
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}

//***************BFS(breadth-first search) code***************//
void bfs(int s,int n)
{
int p,i;

enqueue(s);
vis[s]=1;
p=dequeue();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++){
if((a[p][i]!=0)&&(vis[i]==0))
{
enqueue(i);
vis[i]=1;
}}
p=dequeue();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++){
if(vis[i]==0)
bfs(i,n);}
}
void enqueue(int item)
{

if(rear==19)

        printf("QUEUE FULL");

else {
if(rear==-1)
{
        q[++rear]=item;


        front++;
}
else
```

```
        q[++rear]=item;
}
}
int dequeue()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
return(k);
}
}

//****************DFS(depth-first search) code******************//
void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0){
for(i=1;i<=n;i++){
if((a[k][i]!=0)&&(vis[i]==0)){
push(i);
vis[i]=1;
}}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow ");

else
stack[++top]=item;
}
int pop()
{
```

```
int k;
if(top==-1)
return(0);
else
{
k=stack[top--];
return(k);
}
}
```

Output

```
ENTER THE NUMBER VERTICES 3
ENTER 1 IF 1 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 1 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 3 ELSE 0 0
THE ADJACENCY MATRIX IS
0 1 1
1 0 1
1 1 0

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE1
ENTER THE SOURCE VERTEX :1
1 2  3 DO U WANT TO CONTINUE(Y/N) ? y

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE2
ENTER THE SOURCE VERTEX :1
1  3  2 DO U WANT TO CONTINUE(Y/N) ? n
```

Experiment 19 **Date:20/12/2023**

Prim's Algorithm

Aim:

Program to implement Prim's Algorithm for finding the minimum cost spanning tree.

Algorithm:

1.start

2. declare and initialise
vertex_array[MAX],counter,vertex_count=0,row,column,cost_matrix[MAX][MAX],visited[MAX]={0},edge_count=0,count=1,sum_cost=0,min_cost=0,row_no,column_no,vertex1,vertex2
3.input vertex_count
4. for i= 1 to vertex_count do
input vertex_array[counter]
5.for row=1 to vertex_count do
{
for coloumn=1 to vertex_count do
{
input cost_matrix[row][column]
if(cost_matrix[row][column] == 0)then
{
set cost_matrix[row][column] = 999
}
}
}
6.set visited[1]=1
7.set edge_count = vertex_count-1
8.while(count <= edge_count) do
{
for min_cost=999,row=1 to vertex_count do{
for column=1 to vertex_count do{
if(cost_matrix[row][column] < min_cost) then{
if(visited[row] != 0) then{
set min_cost = cost_matrix[row][column]
set vertex1 = row_no = row
set vertex2 = column_no = column}
}
}
}
}
9.if(visited[row_no] == 0 || visited[column_no] ==0) then
{
print count++,vertex_array[vertex1],vertex_array[vertex2],min_cost)
set sum_cost = sum_cost + min_cost

```
            set visited[column_no]=1
            set cost_matrix[vertex1][vertex2] = cost_matrix[vertex2][vertex1] = 999
        }
 10.print sum_cost
 11.stop
```

Program

```
#include<stdio.h>
#define MAX 10
int main(){
        int vertex_array[MAX],counter;
        int vertex_count=0;
        int row,column;
        int cost_matrix[MAX][MAX];
        int visited[MAX]={0};
        int edge_count=0,count=1;
        int sum_cost=0,min_cost=0;
        int row_no,column_no,vertex1,vertex2;
        printf("Total no of vertex :: ");
        scanf("%d",&vertex_count);
        printf("\n-- Enter vertex -- \n\n");
        for(counter=1;counter<=vertex_count;counter++){
                printf("vertex[%d] :: ",counter);
                scanf("%d",&vertex_array[counter]);
        }

        printf("\n--- Enter Cost matrix of size %d x %d  ---
\n\n",vertex_count,vertex_count);
        printf("\n\t-- format is --\n");
        for(row=1;row<=vertex_count;row++){
                for(column=1;column<=vertex_count;column++){
                        printf("x ");
                }
                printf("\n");
        }
        printf("\n-- MATRIX --\n\n");
        //Get edge weight matrix from user
        for(row=1;row<=vertex_count;row++){
        for(column=1;column<=vertex_count;column++){
                        scanf("%d",&cost_matrix[row][column]);
                        if(cost_matrix[row][column] == 0){
                                cost_matrix[row][column] = 999;}
                }
        }
```

```
        printf("\n");
        visited[1]=1;
        edge_count = vertex_count-1;
        while(count <= edge_count){
                for(row=1,min_cost=999;row<=vertex_count;row++){
                        for(column=1;column<=vertex_count;column++){
                                if(cost_matrix[row][column] < min_cost){
                                        if(visited[row] != 0){
                                                min_cost = cost_matrix[row][column];
                                                vertex1 = row_no = row;
                                                vertex2 = column_no = column;
                                        }
                                }
                        }
                }

                if(visited[row_no] == 0 || visited[column_no] ==0){
                        printf("\nEdge %d is (%d -> %d) with cost : %d
",count++,vertex_array[vertex1],vertex_array[vertex2],min_cost);
                        sum_cost = sum_cost + min_cost;
                        visited[column_no]=1;
                }
                cost_matrix[vertex1][vertex2] = cost_matrix[vertex2][vertex1] = 999;
        }
        printf("\n\nMinimum cost=%d",sum_cost);
        return 0;
}
```

Output

```
Total no of vertex :: 3

-- Enter vertex --
vertex[1] :: 1
vertex[2] :: 2
vertex[3] :: 3

--- Enter Cost matrix of size 3 x 3  ---
-- format is --

        x x x

        x x x

        x x x
```

```
-- MATRIX --
        0 5 7
        5 0 1
        7 1 0

Edge 1 is (1 -> 2) with cost : 5
Edge 2 is (2 -> 3) with cost : 1
Minimum cost=6
```

Experiment 20 **Date:21/12/2023**

Kruskal's Algorithm

Aim:

Program to implement Kruskal's algorithm..

Algorithm:

main()

```
1.start
2.declare and initialize i,j,k,a,b,u,v,n,ne=1,min,mincost=0,cost[9][9],parent[9]
3.input n
4.for i=1 to n do
  {
        for j=1 to n do
        {
              input cost[i][j]
              if(cost[i][j]==0)then
                 set cost[i][j]=999
        }
  }
5.while(ne<n) do
  {
        for(i=1,min=999;i<=n;i++)
        {
              for(j=1;j<=n;j++)
              {
                if(cost[i][j]<min)
                {
                      set min=cost[i][j]
                      set a=u=i
                      set b=v=j
                }
              }
        }
  set u=find(u)
  set v=find(v)
  if(uni(u,v))then
  {
   print ne++,a,b,min
   set mincost +=min
  }
  set cost[a][b]=cost[b][a]=999;
  }
6.print mincost
7.stop
```

int find(int i)

1.start
2.while(parent[i])do
{
set i=parent[i]
}
3.return i
4.exit

int uni(int i,int j)

1.start
2.if(i!=j)then
{
set parent[j]=i;
return 1
}
3.return 0
4.exit

Program

```
#include<stdio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
{
 printf("\nEnter the no. of vertices:");
scanf("%d",&n);
 printf("\nEnter the cost adjacency matrix\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&cost[i][j]);
if(cost[i][j]==0)
cost[i][j]=999;
}
}
printf("\nThe edges of Minimum Cost Spanning Tree are\n");
while(ne<n)
{
for(i=1,min=999;i<=n;i++)
{
for(j=1;j<=n;j++){
```

```
                if(cost[i][j]<min)
                {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                }
            }
        }
        u=find(u);
        v=find(v);
         if(uni(u,v))
        {
         printf("\n%d edge (%d,%d) =%d\n",ne++,a,b,min);
        mincost +=min;
        }
        cost[a][b]=cost[b][a]=999;
        }
         printf("\n\tMinimum cost = %d\n",mincost);
}
int find(int i)
{
        while(parent[i])
                i=parent[i];
        return i;
}

int uni(int i,int j)
{
    if(i!=j)
    {
        parent[j]=i;
        return 1;
    }
    return 0;
}
```

Output

```
Enter the no. of vertices:5
Enter the cost adjacency matrix
0 0 3 0 0
0 0 10 4 0
3 10 0 2 6
0 4 2 0 1
0 0 6 1 0
```

The edges of Minimum Cost Spanning Tree are
1 edge (4,5) =1
2 edge (3,4) =2
3 edge (1,3) =3
4 edge (2,4) =4
Minimum cost = 10

Experiment 21 **Date:04/01/2024**

Disjoint Set Operations

Aim:

Program to perform disjoint set operations create union and find.

Algorithm:

```
struct node
{
        declare struct node *rep,struct node *next,data
}
```

main()

1.start

2.declare struct node*heads[50],*tails[50]

3.declare and initialize countRoot=0,choice,x,i,j,y,flag=0

4.repeat step 5 till choice not equal to 5
5.input choice
a.if(choice==1)then
{
input x
if(search(x)==1)then
print Element already present in the disjoint set DS
else then
call makeSet(x)
}
b.if(choice==2)then
{
for i=0 to i<countRoot do
print heads[i]->data
}
c.if(choice==3)then
{
input x
input y
call unionSets(x,y)
}
d.if(choice==4)then
{
input x
create a node rep dynamically
set rep=find(x)
if(rep==NULL)then
print Element not present in the DS

```
                else then
                        print rep->data
    }
  6.stop
```

void unionSets(int a,int b)

```
  1.start
  2.declare and initialize i,pos,flag=0,j,struct node *rep1=find(a)
  3.create a node tail2 dynamically
  4. set struct node *rep2=find(b)
  5.if(rep1==NULL||rep2==NULL)then
    {
        print Element not present in the DS
        return;
    }
  6.if(rep1!=rep2)then
   {
        for j=0 to j<countRoot do
        {
        if(heads[j]==rep2)then
        {
                set pos=j
                set flag=1
                set countRoot-=1
                set tail2=tails[j]
                for i=pos to countRoot do
                {
                        set heads[i]=heads[i+1]
                        set tails[i]=tails[i+1]
                }
         }

        if(flag==1)then
                break
        }
        for j=0 to j<countRoot do
        {
        if(heads[j]==rep1)then
        {
                set tails[j]->next=rep2
                set tails[j]=tail2
                break
        }
        }
        while(rep2!=NULL)do
        {
        set rep2->rep=rep1
```

```
            set rep2=rep2->next
        }
        call displaySet(rep1)
        }
        7.exit
```

struct node* find(int a)

```
        1.start
        2.declare i
        3.create a node tmp dynamically
        4.for i=0 to countRoot do
          {
            set tmp=heads[i]
            while(tmp!=NULL)do
            {
                if(tmp->data==a)then
                    return tmp->rep
                tmp=tmp->next
            }
            return NULL
          }
        5.exit
```

void displaySet(struct node *rep)

```
        1.start
        2.while (rep != NULL)do
          {
            print rep->data
            set rep = rep->next
          }
        3.exit
```

int search(int x)

```
        1.start
        2.declare i
        3.create a node tmp dynamically
        4.for i=0 to countRoot do
          {
            set tmp=heads[i]
            if(heads[i]->data==x)then
                return 1
            while(tmp!=NULL)do
            {
                if(tmp->data==x)then
                    return 1
                tmp=tmp->next
            }
```

```
  }
5.return 0
6.exit
```

void makeSet(int x)

```
1.start
2.create a node new dynamically
3.set new->rep=new
4.set new->next=NULL
5.set new->data=x
6.set heads[countRoot]=new
7.set tails[countRoot++]=new
```

Program

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node{
   struct node *rep;
   struct node *next;
   int data;
}*heads[50],*tails[50];
static int countRoot=0;
void makeSet(int x){
      struct node *new=(struct node *)malloc(sizeof(struct node));
      new->rep=new;
      new->next=NULL;
      new->data=x;
      heads[countRoot]=new;
      tails[countRoot++]=new;
}
struct node* find(int a){
      int i;
      struct node *tmp=(struct node *)malloc(sizeof(struct node));
      for(i=0;i<countRoot;i++){
            tmp=heads[i];
            while(tmp!=NULL){
            if(tmp->data==a)
            return tmp->rep;
            tmp=tmp->next;
            }
      }
      return NULL;
}
void unionSets(int a,int b){
      int i,pos,flag=0,j;
```

```
        struct node *tail2=(struct node *)malloc(sizeof(struct node));
        struct node *rep1=find(a);

        struct node *rep2=find(b);
        if(rep1==NULL||rep2==NULL){
                printf("\nElement not present in the DS\n");
                return;
        }
        if(rep1!=rep2){
                for(j=0;j<countRoot;j++){
                        if(heads[j]==rep2){
                                pos=j;
                                flag=1;
                                countRoot-=1;
                                tail2=tails[j];
                                for(i=pos;i<countRoot;i++){
                                        heads[i]=heads[i+1];
                                        tails[i]=tails[i+1];
                                }
                        }
                        if(flag==1)
                                break;
                }
                for(j=0;j<countRoot;j++){
                        if(heads[j]==rep1){
                                tails[j]->next=rep2;
                                tails[j]=tail2;
                                break;
                        }
                }
                while(rep2!=NULL){
                rep2->rep=rep1;
                rep2=rep2->next;
                }
        displaySet(rep1);
        }
 }
 void displaySet(struct node *rep) {
    printf("Unioned Set: ");
    while (rep != NULL) {
        printf("%d ", rep->data);
        rep = rep->next;
    }
    printf("\n");
 }
 int search(int x){
        int i;
        struct node *tmp=(struct node *)malloc(sizeof(struct node));
```

```
	for(i=0;i<countRoot;i++){
		tmp=heads[i];
		if(heads[i]->data==x)
			return 1;
		while(tmp!=NULL){
			if(tmp->data==x)
				return 1;
			tmp=tmp->next;
		}
	}

	return 0;
}
void main(){
int choice,x,i,j,y,flag=0;

	do{
		printf("\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n");
		printf("\n.......MENU.......\n\n1.Make Set\n2.Display set
representatives\n3.Union\n4.Find Set\n5.Exit\n");
		printf("Enter your choice : ");
		scanf("%d",&choice);
		printf("\n||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\n");
		switch(choice){
		case 1:
			printf("\nEnter new element : ");
			scanf("%d",&x);
			if(search(x)==1)
				printf("\nElement already present in the disjoint set DS\n");
			else
				makeSet(x);
			break;
		case 2:
			printf("\n");
			for(i=0;i<countRoot;i++)
				printf("%d ",heads[i]->data);
			printf("\n");
			break;
		case 3:
			printf("\nEnter first element : ");
			scanf("%d",&x);
			printf("\nEnter second element : ");
			scanf("%d",&y);
			unionSets(x,y);
			break;
		case 4:
			printf("\nEnter the element");
			scanf("%d",&x);
```

```
                        struct node *rep=(struct node *)malloc(sizeof(struct node));
                        rep=find(x);
                        if(rep==NULL)
                        printf("\nElement not present in the DS\n");
                        else
                        printf("\nThe representative of %d is %d\n",x,rep->data);
                        break;
                case 5:
                        exit(0);
                default:
                        printf("\nWrong choice\n");

                        break;
                }
        }
        while(1);
}
```

Output

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
.......MENU.......

1. Make Set
2. Display set representatives
3. Union
4. Find Set
5. Exit

Enter your choice: 1
Enter new element: 5

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
.......MENU.......

1. Make Set
2. Display set representatives
3. Union
4. Find Set
5. Exit

Enter your choice: 1
Enter new element: 8
```

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
.......MENU.......

1. Make Set
2. Display set representatives
3. Union
4. Find Set
5. Exit

Enter your choice: 2
5 8
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
.......MENU.......

1. Make Set
2. Display set representatives
3. Union
4. Find Set
5. Exit

Enter your choice: 3
Enter first element: 5
Enter second element: 8
Unioned Set:5 8
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
.......MENU.......

1. Make Set
2. Display set representatives
3. Union
4. Find Set
5. Exit

Enter your choice: 4
Enter the element: 8
The representative of 8 is 5

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
.......MENU.......

1. Make Set
2. Display set representatives
3. Union
4. Find Set
5. Exit

Enter your choice: 5
```

Experiment 22 **Date:05/01/2024**

Dijkstras Algorithm

Aim:

Program for single source shortest path algorithm using Dijkstras algorithm

Algorithm:

main()

1.start
2.declare and initialize INFINITY=9999,MAX=10,G[MAX][MAX],i,j,n,u
3.input n
4.for i=0 to n do
{
for j=0 to j<n do
{
input G[i][j]
}
}

5.input u
6.call dijkstra(G,n,u)
7.stop

void dijkstra(int G[MAX][MAX],int n,int startnode)

1.start
2.declare cost[MAX][MAX],distance[MAX],pred[MAX], visited[MAX],
count,mindistance,nextnode,i,j
3.for i=0 to n do
{
for j=0 to n do
{
if(G[i][j]==0)then
set cost[i][j]=INFINITY
else then
set cost[i][j]=G[i][j]
}
}

4.for i=0 to n do
{
set distance[i]=cost[startnode][i]
set pred[i]=startnode
set visited[i]=0

```
   }
5.set distance[startnode]=0
6.set visited[startnode]=1
7.set count=1
8.while(count<n-1) do
  {
            set mindistance=INFINITY
            for i=0 to n do
            {
            if(distance[i]<mindistance&&!visited[i])then
            {
                  set mindistance=distance[i]
                  set nextnode=i
            }
            set visited[nextnode]=1
            for i=0 to n do
            {
                  if(!visited[i])then
                  {
                        if(mindistance+cost[nextnode][i]<distance[i])then
                        {
                          set distance[i]=mindistance+cost[nextnode][i]
                          set pred[i]=nextnode
                        }
                  }
            }
            count++;
  }

9.for i= 0 to n do
 {
            if(i!=startnode)then
            {
                  print distance[i]
                  print i
                  set j=i

             while(j!=startnode)do
             {
                   set j=pred[j]
                   print j
             }
   }
10.exit
```

Program

```
#include<stdio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main() {
	int G[MAX][MAX],i,j,n,u;
	printf("Enter no. of vertices:");
	scanf("%d",&n);
	printf("\nEnter the adjacency matrix:\n");

	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			scanf("%d",&G[i][j]);

	printf("\nEnter the starting node:");
	scanf("%d",&u);
	dijkstra(G,n,u);
}

void dijkstra(int G[MAX][MAX],int n,int startnode){
	int cost[MAX][MAX],distance[MAX],pred[MAX];
	int visited[MAX],count,mindistance,nextnode,i,j;

	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			if(G[i][j]==0)
			cost[i][j]=INFINITY;
			else cost[i][j]=G[i][j];


	for(i=0;i<n;i++){
		distance[i]=cost[startnode][i];
		pred[i]=startnode; visited[i]=0;
	 }

	 distance[startnode]=0;
	 visited[startnode]=1;
	 count=1;

	 while(count<n-1){
	  mindistance=INFINITY;
	  for(i=0;i<n;i++)
	  if(distance[i]<mindistance&&!visited[i]) {
```

```
                    mindistance=distance[i];
                    nextnode=i;
            }
            visited[nextnode]=1;
        for(i=0;i<n;i++)
                if(!visited[i])
                    if(mindistance+cost[nextnode][i]<distance[i]){
                        distance[i]=mindistance+cost[nextnode][i];
                            pred[i]=nextnode;
                        }
          count++;
        }


        for(i=0;i<n;i++ )
                if(i!=startnode){
                    printf("\nDistance of node %d = %d",i,distance[i]);
                    printf("\nPath = %d",i); j=i ;

                 do{
                    j=pred[j];
                     printf(" <- %d",j);
                  }while(j!=startnode);
                }
}
```

Output

```
Enter no. of vertices: 5
Enter the adjacency matrix:
  0 10 5 0 0
  0 0 2 1 0
  0 3 0 9 2
  0 0 0 0 4
  7 0 0 6 0
Enter the starting node: 0
Distance of node 1 = 8
Path = 1 <- 2 <- 0
Distance of node 2 = 5
Path = 2 <- 0
Distance of node 3 = 9
Path = 3 <- 2 <- 0
Distance of node 4 = 7
Path = 4 <- 2 <- 0
```