GURU NANAK DEV 1956 ENGG. COLLEGE
(Punjab Govt. Aided Status)

ਗੁਰੂ ਨਾਨਕ ਦੇਵ ਇੰਜੀਨੀਅਰਿੰਗ ਕਾਲਿਜ,ਲੁਧਿਆਣਾ
Guru Nanak Dev Engineering College, Ludhiana
An Autonomous College U/s UGC Act - 1956 [2(f) and 12(B)]
AICTE Approved, ISO 9001:2015 CERTIFIED
Affliated to I. K. Gujral Punjab Technical University, Jalandhar
IEI ACCREDITED UG PROGRAMMES
Institute Accredited by NAAC (A Grade) & TCS.

PRACTICAL FILE

DATA STRUCTURE LABORATORY

NAME – Jaswant Singh
URN – 1905009
CRN – 1915040
CLASS – D2CSA2
SUBMITTED TO - Prof. Jaswant Singh

PROGRAMMING LANGUAGES USED – C, C++
GITHUB REPO - Link to github repo

INDEX

S. No.	PRACTICAL NAME
1.	Design, Develop and Implement a menu driven Program for the following Array operations a.Creating an Array of N Integer Elements b.Display of Array Elements with Suitable Headings c.Inserting an Element (ELEM) at a given valid Position(POS) d.Deleting an Element at a given validPosition(POS) e.Exit.
2.	Design, Develop and Implement a menu driven Program for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) a.Push an Element on to Stack b.Pop an Element from Stack c.Demonstrate how Stack can be used to check Palindrome d.Demonstrate Overflow and Underflow situations on Stack Display the status of Stack f. Exit Support the program with appropriate functions for each of the above operations
3.	Design,Develop and Implement a Program for converting an Infix Expressionto Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder),^(Power) and alphanumeric operands.
4.	Design, Develop and Implement a Program for the following Stack Applicationsa.Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %,^b.Solving Tower of Hanoi problem with ndisks
5.	Design, Develop and Implement a menu driven Program for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX) a.Insert an Element on to Circular QUEUE b.Delete an Element from Circular QUEUE c.Demonstrate Overflow and Underflow situations on Circular QUEUE d.Display the status of Circular QUEUE e. Exit Support the program with appropriate functions for each of the above operations .
6.	Design, Develop and Implement a menu driven Program for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo a.Create a SLL of N Students Data by using front insertion. b.Display the status of SLL and count the number of nodes in it c.Perform Insertion / Deletion at End ofSLL d.Perform Insertion / Deletion at Front of SLL(Demonstration ofstack) e.Exit
7.	Design, Develop andImplement a menu driven Program for the following operations on Doubly Linked List (DLL) of Employee Data

	with the fields: SSN, Name, Dept,Designation, Sal, PhNo a.Create a DLL of N Employees Data by using end insertion. b.Display the status of DLL and count the number of nodes in it c.Perform Insertion and Deletion at End of DLL d.Perform Insertion and Deletion at Front of DLL e.Demonstrate how this DLL can be used as Double Ended Queue f.Exit
8.	Design, Develop and Implement a Program for the following operationsonSingly Circular Linked List (SCLL) with header nodes a. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)
9.	Design, Develop and Implement a menu driven Program for the following operations on Binary Search Tree (BST) of Integers a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 b. Traverse the BST in Inorder, Preorder and Post Order c. Search the BST for a given element (KEY) and report the appropriate messagee. Exit
10.	Design, Develop and Implement a Program for the following operations on Graph(G) of Cities a.Create a Graph of N cities using Adjacency Matrix. b.Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method
11.	Write a Program to findsthe position of an element in an array using Linear SearchAlgorithm and Binary search Algorithm.
12.	Write a program to sort list using different sorting algorithms (bubble, selection, insertion,radix, merge and quick sort) and compare them.

PRACTICAL – 1

Design, Develop and Implement a menu driven Program for the following Array operations

a. Creating an Array of N Integer Elements

b. Display of Array Elements with Suitable Headings

c. Inserting an Element (ELEM) at a given valid Position (POS)

d. Deleting an Element at a given valid Position(POS)

e. Exit.

PROGRAM:

```
#include<bits/stdc++.h>
using namespace std;
int arr[100],n;
int create();
int display();
int insert();
int del();
int main()
{
int num;
while(true)
{
cout << "\n\t\t$$$ MENU $$$\n1 => Creating an Array of N Integer Elements\n2 => Display of
Array Elements with Suitable Headings\n3 => Inserting an Element (ELEM) at a given valid
Position (POS)\n4 => Deleting an Element at a given valid Position(POS)\n5 => Exit\n ";
cout << "\n\n Enter a number from the menu.\n";
cin >> num;
switch(num)
{
case 1:
cout << "Creating an array of n elements......\n";
create();
break;
case 2:
cout << "Displaying elements of array......\n";
display();
break;
case 3:
insert();
break;
case 4:
del();
break;
case 5:
cout << "\nExiting....!!!!\n";
return 0;
default:
cout << "\nERROR..!!! Invalid option entered\n";
}
}
return 0;
}
int create()
{
cout << "Enter the wanted size of array\n";
cin >> n;
for(int i=0;i<n;++i)
{
cin>>arr[i];
}
return 0;
}
int display()
{
char c;
for(int i=0;i<n;i++)
{
cout <<"Element "<<i+1<<" = "<<arr[i]<<"\n";
}
cout<<"\nPress any key to continue...!!!\n";
cin >> c;
return 0;
}
int insert()
{
int num,index;
cout<< "Enter the index where number is to be inserted\n";
cin>>index;
```

```
cout<<"Enter the number to be inserted\n";
cin >> num;
if(index>n+1)
{
cout<<"ERROR!!! index greater than size of array is not allowed.";
return 0;
}
for(int i =n-1;i>=index-1;i--)
{
arr[i+1]=arr[i];
}
arr[index-1]=num;
++n;
display();
return 0;
}
int del()
{
int index;
cout << "Enter the index to be deleted\n";
cin >> index;
if(index>n+1)
{
cout << "ERROR!!! Enter a valid position.";
}
for(int i=index-1;i<n;++i)
{
arr[i]=arr[i+1];
}
--n;
display();
return 0;
}
```

OUTPUT :

```
$$$ MENU $$$

1 => Creating an Array of N Integer Elements

2 => Display of Array Elements with Suitable Headings

3 => Inserting an Element (ELEM) at a given valid Position (POS)
```

```
4 => Deleting an Element at a given valid Position(POS)

5 => Exit

Enter a number from the menu.

1

Creating an array of n elements......

Enter the wanted size of array

5

1

2

4

5

6$$$ MENU $$$

1 => Creating an Array of N Integer Elements

2 => Display of Array Elements with Suitable Headings

3 => Inserting an Element (ELEM) at a given valid Position (POS)

4 => Deleting an Element at a given valid Position(POS)

5 => Exit

Enter a number from the menu.

3

Enter the index where number is to be inserted

3

Enter the number to be inserted

3

Element 1 = 1

Element 2 = 2

Element 3 = 3

Element 4 = 4

Element 5 = 5

Element 6 = 6
```

```
Press any key to continue...!!!
4
$$$ MENU $$$
1 => Creating an Array of N Integer Elements
2 => Display of Array Elements with Suitable Headings
3 => Inserting an Element (ELEM) at a given valid Position (POS)
4 => Deleting an Element at a given valid Position(POS)
5 => ExitEnter a number from the menu.
4
Enter the index to be deleted
6
Element 1 = 1
Element 2 = 2
Element 3 = 3
Element 4 = 4
Element 5 = 5
Press any key to continue...!!!
5
$$$ MENU $$$
1 => Creating an Array of N Integer Elements
2 => Display of Array Elements with Suitable Headings
3 => Inserting an Element (ELEM) at a given valid Position (POS)
4 => Deleting an Element at a given valid Position(POS)
5 => Exit
Enter a number from the menu.
5
Exiting....!!!!
```

PRACTICAL – 2

Design, Develop and Implement a menu driven Program for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

a. Push an Element on to Stack

b. Pop an Element from Stack

c. Demonstrate how Stack can be used to check Palindrome

d. Demonstrate Overflow and Underflow situations on Stack Display the status of Stack

e. Exit

Support the program with appropriate functions for each of the above operations.

PROGRAM:

```
#include<bits/stdc++.h>
#define MAX 100
using namespace std;
int stck[MAX], n=100, top=-1;
int push();
int pop();
int palindrome();
int overunder();
int display();
int main()
{
while(1)
{
int n;
cout << "\n\n####### MENU #######\n";
cout << "1 => Push an Element on to Stack.\n2 => Pop an Element from Stack.\n3 => Demonstrate
how Stack can be used to check Palindrome.\n4 => Demonstrate Overflow and Underflow
situations on Stack Display the status of Stack\n5 => Exit \n\n";
cout << "\nEnter a number from the menu\n";
cin >> n;
switch(n)
{
case 1:
push();
break;
case 2:
pop();
break;
case 3:
palindrome();
break;
case 4:
overunder();
break;
case 5:
return 0;
default:
cout << "ERROR...!!! Enter a valid value\n";
}
}
return 0;
}
int push()
{
int num;
cout << "\nEnter the number to be pushed\n";
cin >> num;
if(top>=n-1) cout<<"Stack Overflow\n"<<endl;
else
{
top++;
stck[top]=num;
}
display();
return 0;
}
int pop()
```

```
{
cout << "Popping element from stack....!!!\n";
if(top==-1)
{
cout <<"ERROR Stack Underflow\n";
}
else
{
cout<<"The popped element is "<< stck[top] <<endl;--top;
}
display();
return 0;
}
int palindrome()
{
if(top==-1)
{
cout<<"Enter some elements\n";
return 0;
}
int arr[MAX];
for(int i=0;i<MAX;++i)
{
arr[i]=stck[i];
}
int t = top;
int c=0,d=0;
for(int i=0;i<t;++i)
{
if(arr[t-i-1]==stck[i])
{
++c;
}
++d;
}
if(c==d)
{
cout << "WOW...!!! PALINDROME\n";
}
else
{
cout << "OOPS...!!! NOT A PALINDROME\n";
}
display();
return 0;
}
int overunder()
{
if(top==-1)
{
cout<<"Stack Underflow\n";
}
else if(top>n-1)
{
cout << "Stack Overflow\n";
}
else
{
cout << "Space is available in stack\n";
}
display();
return 0;
}
int display()
{
int t=top;
cout << "The elements in stack are:\n";
while(t>=0)
{
cout << stck[t]<<"\n";
--t;
if(t==-1)
{
break;
}
}
char c;
cout<<"Press a key and hit enter to continue\n";
cin >> c;
return 0;
}
```

OUTPUT :

```
####### MENU #######
```

```
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
1
Enter the number to be pushed
1
The elements in stack are:
1
Press a key and hit enter to continue
2
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
1
Enter the number to be pushed
2
The elements in stack are:
2
1
Press a key and hit enter to continue
```

```
d
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu1
Enter the number to be pushed
1
The elements in stack are:
1
2
1
Press a key and hit enter to continue
c
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
1
Enter the number to be pushed
3
The elements in stack are:
```

```
3
1
2
1
Press a key and hit enter to continue
a
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
2
Popping element from stack....!!!
The popped element is 3
The elements in stack are:
1
2
1
Press a key and hit enter to continue
r
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.4 => Demonstrate Overflow
and Underflow situations on Stack Display the
status of Stack
5 => Exit
```

```
Enter a number from the menu
3
WOW...!!! PALINDROME
The elements in stack are:
1
2
1
Press a key and hit enter to continue
g
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
4
Space is available in stack
The elements in stack are:
1
2
1
Press a key and hit enter to continue
r
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
```

```
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
2
Popping element from stack....!!!
The popped element is 1
The elements in stack are:
2
1
Press a key and hit enter to continue
f
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.4 => Demonstrate Overflow
and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
2
Popping element from stack....!!!
The popped element is 2
The elements in stack are:
1
Press a key and hit enter to continue
f
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
```

```
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
2
Popping element from stack....!!!
The popped element is 1
The elements in stack are:
Press a key and hit enter to continue
b
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
status of Stack
5 => Exit
Enter a number from the menu
4
Stack Underflow
The elements in stack are:
Press a key and hit enter to continue
r
####### MENU #######
1 => Push an Element on to Stack.
2 => Pop an Element from Stack.
3 => Demonstrate how Stack can be used to check Palindrome.
4 => Demonstrate Overflow and Underflow situations on Stack Display the
```

```
status of Stack
5 => Exit
Enter a number from the menu
5
```

PRACTICAL – 3

Design, Develop and Implement a Program for converting an Infix Expression to Postfix Expression.

Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^(Power) and alphanumeric operands.

PROGRAM:

```
#include<bits/stdc++.h>
using namespace std;
int convert(string);
int precedence(char);
int main()
{
string s;
cout << "Enter the infix statement\n";
cin >> s;
convert(s);
int c=1;
while(c==1)
{
cout << "To convert one more infix expression to postfix, press 1 else press 0 to exit and hit enter\n
";
cin >> c;
if(c==0)
{
return 0;
}
cout << "Enter the infix statement\n";
cin >> s;
convert(s);
}
return 0;
}
```

```
int convert(string s)
{
stack<char> stck;
string dup;
int len;
//stck.push('e');
len=s.length();
for(int i=0;i<len;++i)
{
if((s[i]>='A'&&s[i]<='Z') || (s[i]>='a'&&s[i]<='z'))
{
dup=dup+s[i];
}
else if(s[i]=='(')
{
stck.push('(');
}
else if(s[i]==')')
{
while(stck.top()!='('&& stck.empty() == false)
{
char ch = stck.top();
stck.pop();
dup =dup + ch;
}
if(stck.top() == '(')
{
char ch = stck.top();
stck.pop();
}
}
else
{
while(stck.empty() == false&& precedence(s[i]) <= precedence(stck.top()))
{
char ch = stck.top();
stck.pop();
dup = dup + ch;
}
stck.push(s[i]);
}
}
while(stck.empty() == false)
{
char ch = stck.top();
stck.pop();
dup = dup + ch;
}
cout << "POSTIX EXPRESSION IS: ";
cout << dup <<"\n";
return 0;
}
int precedence(char ch)
{
if(ch == '^')
return 3;
else if(ch == '*'|| ch == '/')
return 2;
else if(ch == '+'|| ch == '-')
return 1;
else return -1;
}
```

OUTPUT :

```
Enter the infix statement
a+b
POSTIX EXPRESSION IS: ab+
To convert one more infix expression to postfix, press 1 else press 0 to
exit and hit enter
1
Enter the infix statement
a+b*(c^d-e)^(f+g*h)-i
POSTIX EXPRESSION IS: abcd^e-fgh*+^*+i-
To convert one more infix expression to postfix, press 1 else press 0 to
exit and hit enter
```

PRACTICAL – 4

Design, Develop and Implement a Program for the following Stack Applications

a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

b. Solving Tower of Hanoi problem with n disks.

a.) PROGRAM:-

```
#include<bits/stdc++.h>
using namespace std;
//function to compute calculation between operands and operator
float compute(char symbol, float op1, float op2)
{
switch (symbol)
{
case '+': return op1 + op2;
case '-': return op1 - op2;
case '*': return op1 * op2;
case '/': return op1 / op2;
case '$':
case '^': return pow(op1,op2);
default : return 0;
}
}
//Driver function
int main()
{
```

```
float s[20], res, op1, op2;
int top, i;
// initialised a char array to make stack
char postfix[20], symbol;
cout << "\nEnter the postfix expression:\n";
// input postfix expression
cin >> postfix;
// top of stack is initialised to -1 which means stack is empty
top=-1;
// traversing and checking comdition for calculations
for (i=0; i<strlen(postfix) ;i++)
{
symbol = postfix[i];
if(isdigit(symbol))
s[++top]=symbol - '0';
else
{
op2 = s[top--];
op1 = s[top--];
// calling compute function and storing return value in res variable
res = compute(symbol, op1, op2);
s[++top] = res;
}
}
res = s[top--];
// output of answer
cout << "\nThe result is : "<<res<<endl;
}
```

OUTPUT :

```
Enter the postfix expression:
12345*+*+
The result is : 47
```

b.) PROGRAM:

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

```
#include <math.h>
#include <stdlib.h>
#include <limits.h>

// A structure to represent a stack
struct Stack
{
unsigned capacity;
int top;
int *array;
};

// function to create a stack of given capacity.
struct Stack* createStack(unsigned capacity)
{
	struct Stack* stack =
		(struct Stack*) malloc(sizeof(struct Stack));
	stack -> capacity = capacity;
	stack -> top = -1;
	stack -> array =
		(int*) malloc(stack -> capacity * sizeof(int));
	return stack;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
return (stack->top == stack->capacity - 1);
}

// Stack is empty when top is equal to -1
int isEmpty(struct Stack* stack)
{
return (stack->top == -1);
}

// Function to add an item to stack. It increases
// top by 1
void push(struct Stack *stack, int item)
{
	if (isFull(stack))
		return;
	stack -> array[++stack -> top] = item;
}

// Function to remove an item from stack. It
// decreases top by 1
int pop(struct Stack* stack)
{
	if (isEmpty(stack))
		return INT_MIN;
	return stack -> array[stack -> top--];
}

//Function to show the movement of disks
void moveDisk(char fromPeg, char toPeg, int disk)
{
	printf("Move the disk %d from \'%c\' to \'%c\'\n",
		disk, fromPeg, toPeg);
}

// Function to implement legal movement between
// two poles
void moveDisksBetweenTwoPoles(struct Stack *src,
                  struct Stack *dest, char s, char d)
{
      int pole1TopDisk = pop(src);
      int pole2TopDisk = pop(dest);

      // When pole 1 is empty
      if (pole1TopDisk == INT_MIN)
      {
            push(src, pole2TopDisk);
            moveDisk(d, s, pole2TopDisk);
      }

      // When pole2 pole is empty
      else if (pole2TopDisk == INT_MIN)
      {
            push(dest, pole1TopDisk);
            moveDisk(s, d, pole1TopDisk);
      }

      // When top disk of pole1 > top disk of pole2
      else if (pole1TopDisk > pole2TopDisk)
      {
            push(src, pole1TopDisk);
            push(src, pole2TopDisk);
            moveDisk(d, s, pole2TopDisk);
      }

      // When top disk of pole1 < top disk of pole2
      else
      {
            push(dest, pole2TopDisk);
            push(dest, pole1TopDisk);
            moveDisk(s, d, pole1TopDisk);
      }
}

//Function to implement TOH puzzle
void tohIterative(int num_of_disks, struct Stack
                  *src, struct Stack *aux,
                  struct Stack *dest)
{
      int i, total_num_of_moves;
      char s = 'S', d = 'D', a = 'A';

      //If number of disks is even, then interchange
      //destination pole and auxiliary pole
      if (num_of_disks % 2 == 0)
      {
            char temp = d;
            d = a;
            a = temp;
      }
      total_num_of_moves = pow(2, num_of_disks) - 1;

      //Larger disks will be pushed first
      for (i = num_of_disks; i >= 1; i--)
            push(src, i);

      for (i = 1; i <= total_num_of_moves; i++)
      {
            if (i % 3 == 1)
            moveDisksBetweenTwoPoles(src, dest, s, d);

            else if (i % 3 == 2)
            moveDisksBetweenTwoPoles(src, aux, s, a);

            else if (i % 3 == 0)
            moveDisksBetweenTwoPoles(aux, dest, a, d);
      }
}

int main()
{
      unsigned num_of_disks = 3;

      struct Stack *src, *dest, *aux;

      src = createStack(num_of_disks);
      aux = createStack(num_of_disks);
      dest = createStack(num_of_disks);

      tohIterative(num_of_disks, src, aux, dest);
      return 0;
}
```

OUTPUT

```
Move the disk 1 from 'S' to 'D'
Move the disk 2 from 'S' to 'A'
Move the disk 1 from 'D' to 'A'
Move the disk 3 from 'S' to 'D'
Move the disk 1 from 'A' to 'S'
Move the disk 2 from 'A' to 'D'
Move the disk 1 from 'S' to 'D'
```

PRACTICAL 5

Design, Develop and Implement a menu driven Program for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate Overflow and Underflow situations on Circular QUEUE

d. Display the status of Circular QUEUE

e. Exit

Support the program with appropriate functions for each of the above operations .

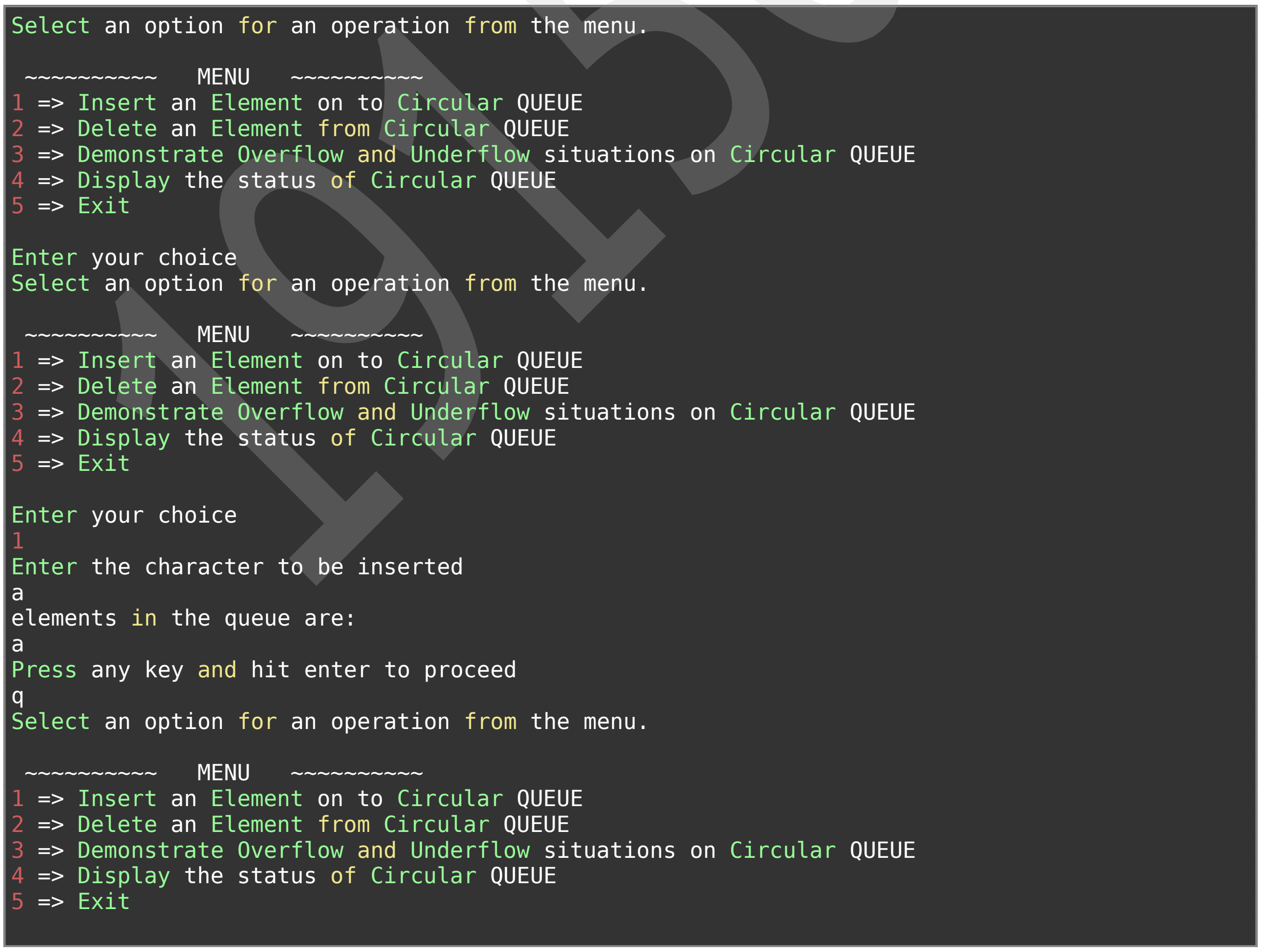
PROGRAM:

```
#include<bits/stdc++.h>
#define MAX 100
using namespace std;
void insert();
void pop();
void status();
void display();
char arr[MAX];
int top=0;
int indx=-1;
int main()
{
cout << "Select an option for an operation from the menu.\n\n ";
cout << "~~~~~~~~~~~ MENU ~~~~~~~~~~~\n";
cout << "1 => Insert an Element on to Circular QUEUE \n";
cout << "2 => Delete an Element from Circular QUEUE \n";
cout << "3 => Demonstrate Overflow and Underflow situations on Circular QUEUE \n";
cout << "4 => Display the status of Circular QUEUE \n";
```

```
cout << "5 => Exit\n\n";
cout << "Enter your choice\n";
while(1)
{
cout << "Select an option for an operation from the menu.\n\n ";
cout << "~~~~~~~~~~~ MENU ~~~~~~~~~~~\n";
cout << "1 => Insert an Element on to Circular QUEUE \n";
cout << "2 => Delete an Element from Circular QUEUE \n";
cout << "3 => Demonstrate Overflow and Underflow situations on Circular QUEUE \n";
cout << "4 => Display the status of Circular QUEUE \n";
cout << "5 => Exit\n\n";
cout << "Enter your choice\n";
int n;
cin >> n;
switch(n)
{
case 1:
insert();
break;
case 2:
pop();
break;
case 3:
status();
break;
case 4:
display();
break;
case 5:
return 0;
default:
cout << "Please Enter a valid option\n";
}
}
return 0;
}
void insert()
{
cout << "Enter the character to be inserted\n";
char c;
```

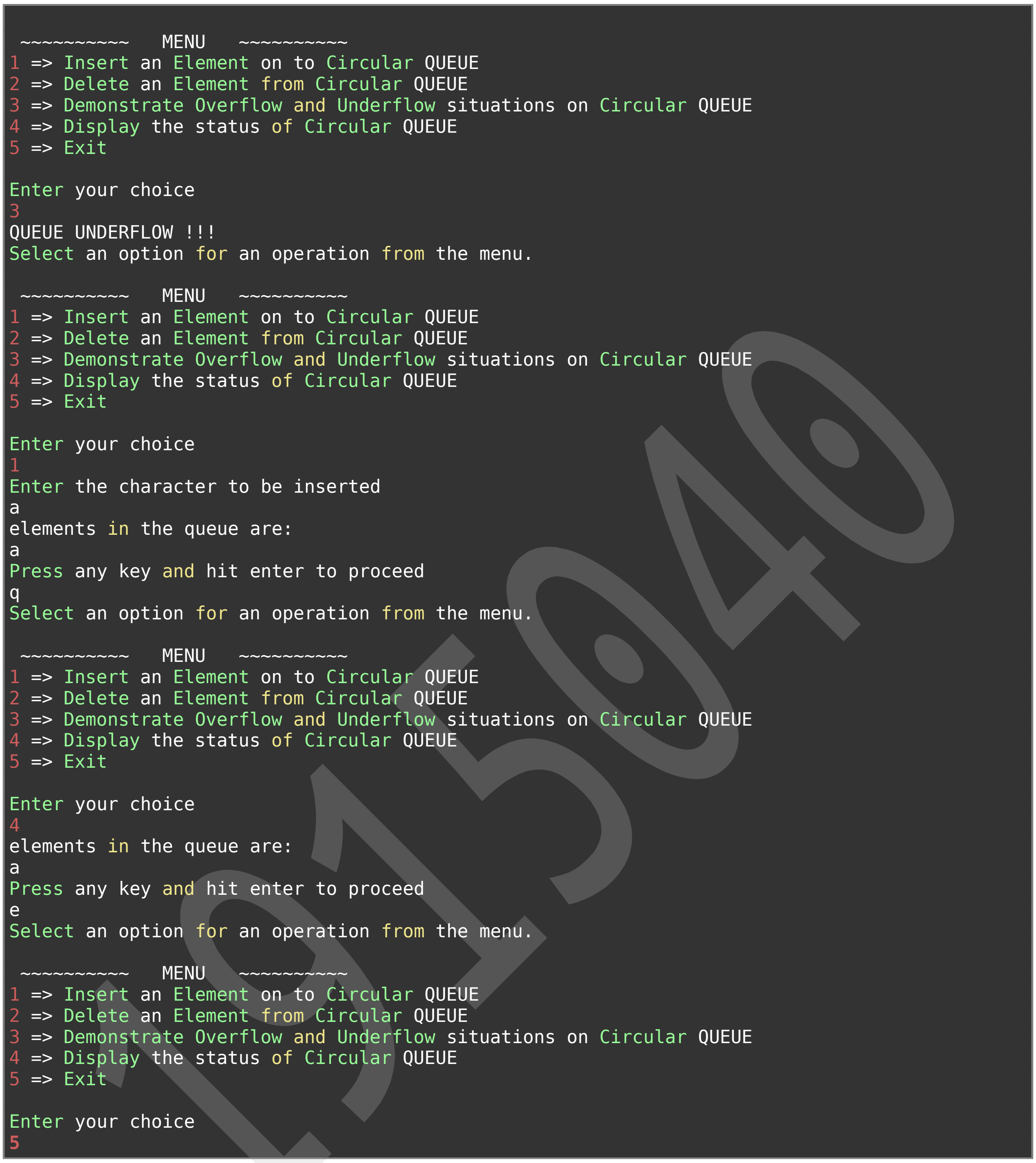
```
cin >> c;
if(indx==-1)
{
arr[indx+1] = c;
indx++;
}
else
{
arr[indx+1]=c;
indx++;
}
display();
}

void pop()
{
if(indx==-1)
{
cout << "Queue is already Empty!!!\n";
}
else
{
if(indx==0)
{
top=-1;
indx=-1;
}
else
{
for(int i=0;i<indx;++i)
{
arr[i]=arr[i+1];
}
indx--;
}
}
display();
}

void status()
{
if(indx==MAX)
{
cout <<"QUEUE OVERFLOW !!!\n";
}
else if(top==-1)
{
cout <<"QUEUE UNDERFLOW !!!\n";
}
}
void display()
{
cout << "elements in the queue are:\n";
for(int i=0;i<=indx;++i)
{
cout << arr[i] <<"\t";
}
cout << "\n";
char c;
cout << "Press any key and hit enter to proceed\n";
cin >> c;
}
```

OUTPUT:



```
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
1
Enter the character to be inserted
a
elements in the queue are:
a
Press any key and hit enter to proceed
q
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit
```

```
Enter your choice
1
Enter the character to be inserted
b
elements in the queue are:
a        b
Press any key and hit enter to proceed
q
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
1
Enter the character to be inserted
c
elements in the queue are:
a        b        c
Press any key and hit enter to proceed
q
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
2
elements in the queue are:
b        c
Press any key and hit enter to proceed
q
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
2
elements in the queue are:
c
Press any key and hit enter to proceed
q
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
2
elements in the queue are:

Press any key and hit enter to proceed
q
Select an option for an operation from the menu.
```



```
 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
3
QUEUE UNDERFLOW !!!
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
1
Enter the character to be inserted
a
elements in the queue are:
a
Press any key and hit enter to proceed
q
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
4
elements in the queue are:
a
Press any key and hit enter to proceed
e
Select an option for an operation from the menu.

 ~~~~~~~~~~   MENU   ~~~~~~~~~~
1 => Insert an Element on to Circular QUEUE
2 => Delete an Element from Circular QUEUE
3 => Demonstrate Overflow and Underflow situations on Circular QUEUE
4 => Display the status of Circular QUEUE
5 => Exit

Enter your choice
5
```

PRACTICAL – 6

Design, Develop and Implement a menu driven Program for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo

a. Create a SLL of N Students Data by using front insertion.

b. Display the status of SLL and count the number of nodes in it

c. Perform Insertion / Deletion at End of SLL

d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)

e. Exit

PROGRAM :

```
#include <bits/stdc++.h>
using namespace std;
int countt=0;
struct stud
{
long long int ph;
int sem;
char name[15],usn[15],brnch[8];
struct stud *next;
}*head=NULL,*tail=NULL,*temp=NULL,*temp1;
void create(long long int n,int s,char na[20],char u[15],char b[5])
{
if(head==NULL)
{
head=(struct stud*)malloc(1*sizeof(struct stud));
head->ph=n;
head->sem=s;
strcpy(head->name,na);
strcpy(head->usn,u);
strcpy(head->brnch,b);
head->next=NULL;
tail=head;
countt++;
}
else
{
temp=(struct stud*)malloc(1*sizeof(struct stud));
temp->ph=n;
temp->sem=s;
strcpy(temp->name,na);
strcpy(temp->usn,u);
strcpy(temp->brnch,b);
temp->next=NULL;
tail->next=temp;
tail=temp;
countt++;
}
}
void display()
{
temp1=head;
if(temp1==NULL)
{
printf("\nlist is empty\n");
}
else
{
printf("student details are as follows:\n");
while(temp1!=NULL)
{
printf(" \n");
printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",temp1 -
>name,temp1->usn,temp1->brnch,temp1->sem,temp1->ph);
printf(" \n");
temp1=temp1->next;

}
printf("no. of nodes=%d\n",countt);
}
}
void insert_head(long long int n,int s,char na[15],char u[15],char b[8])
```

```
{
temp=(struct stud*)malloc(1*sizeof(struct stud));
temp->ph=n;
temp->sem=s;
strcpy(temp->name,na);
strcpy(temp->usn,u);
strcpy(temp->brnch,b);
temp->next=head;
head=temp;
countt++;
}
void insert_tail(long long int n,int s,char na[15],char u[15],char b[8])
{
temp=(struct stud*)malloc(1*sizeof(struct stud));
temp->ph=n;
temp->sem=s;
strcpy(temp->name,na);
strcpy(temp->usn,u);
strcpy(temp->brnch,b);
tail->next=temp;
temp->next=NULL;
tail=temp;
countt++;
}
void delete_head()
{
temp1=head;
if(temp1==NULL)
{
printf("list is empty\n");
}
else
{
head=head->next;
printf("deleted node is:\n");
printf(" \n");
printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",temp1->name,
temp1->usn,temp1->brnch,temp1->sem,temp1->ph);
printf(" \n");
free(temp1);
countt--;
}
}
```

```
void delete_tail()
{
temp1=head;
if(temp1==NULL)
{
printf("list is empty\n");
}
while(temp1->next!=tail)
{
temp1=temp1->next;
}
printf("deleted node is:\n");
printf(" \n");
printf("NAME:%s\nUSN:%s\nBRANCH:%s\nSEM:%d\nPHONE NO.:%lld\n",tail->name,tail-
>usn,tail->brnch,tail->sem,tail->ph);
printf("\n");
free(tail);
tail=temp1;
tail->next=NULL;
countt--;
}
int main()
{
int choice;
long long int ph; int sem;
char name[20],usn[15],brnch[5];
printf("--------MENU \n");
printf("1.create\n2.Insert from head\n3.Insert from tail\n4.Delete from head\5.Delete fromtail\
n6.display \n7.exit\n");
printf(" \n");
while(1)
{
printf("enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the name usn branch sem phno. of the student respectively\n");
scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);
create(ph,sem,name,usn,brnch);
break;
case 2:printf("enter the name usn branch sem phno. of the student respectively\n");
scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);
insert_head(ph,sem,name,usn,brnch);
break;
case 3:printf("enter the name usn branch sem phno. of the student respectively\n");
scanf("%s%s%s%d%lld",name,usn,brnch,&sem,&ph);
insert_tail(ph,sem,name,usn,brnch);
break;
case 4:delete_head();
break;
case 5:delete_tail();
break;
case 6:display();
break;
case 7:exit(0);
default:printf("invalid option\n");
}
}
return 0;
}
```

PRACTICAL – 7

Design, Develop and Implement a menu driven Program for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

a. Create a DLL of N Employees Data by using end insertion.

b. Display the status of DLL and count the number of nodes in it

c. Perform Insertion and Deletion at End of DLL

d. Perform Insertion and Deletion at Front of DLL

e. Demonstrate how this DLL can be used as Double Ended Queue

f. Exit

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Enode
{
char ssn[15];
char name[20];
char dept[5];
char designation[10];
int salary;
long long int phno;
struct Enode *left;
struct Enode *right;
```

```
}*head=NULL;
struct Enode *tail,*temp1,*temp2;
void create(char [],char [],char [],char [],int ,long long int);
void ins_beg(char [],char [],char [],char [],int ,long long int);
void ins_end(char [],char [],char [],char [],int ,long long int);
void del_beg();
void del_end();
void display();
int count=0;
void main()
{
int choice;
char s[15],n[20],dpt[5],des[10];
int sal;
long long int p;
printf("1.Create\n2.Display\n3.Insert at beginning\n4.Insert at End\n5.Delete at
beginning\n6.Delete
at
End\n7.Exit\n");
while(1)
{
printf("\nEnter your choice\n");
scanf("%d",&choice);
switch(choice)
{case 1:printf("Enter the required data(Emp no,Name,Dept,Desig,sal,phone\n)");
scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
create(s,n,dpt,des,sal,p);
break;
case 2:display();
break;
case 3:printf("Enter the required data (Emp no,Name,Dept,Desig,sal,phone\n)");
scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
ins_beg(s,n,dpt,des,sal,p);
break;
case 4:printf("Enter the required data(Emp no,Name,Dept,Desig,sal,phone\n)");
scanf("%s%s%s%s%d%lld",s,n,dpt,des,&sal,&p);
ins_end(s,n,dpt,des,sal,p);
break;
case 5:del_beg();
break;
case 6:del_end();
break;
case 7:exit(0);
}
}
}
void create(char s[15],char n[20],char dpt[5],char des[10],int sal,long long int
p)
{
if(head==NULL)
{
head=(struct Enode *)malloc(1*sizeof(struct Enode));
strcpy(head->ssn,s);
strcpy(head->name,n);
strcpy(head->dept,dpt);
strcpy(head->designation,des);
head->salary=sal;
head->phno=p;
head->left=NULL;
head->right=NULL;
tail=head;
}
else
{
temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
strcpy(temp1->ssn,s);
strcpy(temp1->name,n);
strcpy(temp1->dept,dpt);
strcpy(temp1->designation,des);
temp1->salary=sal;
```

```
temp1->phno=p;
tail->right=temp1;
temp1->right=NULL;
temp1->left=tail;
tail=temp1;
}
}void display()
{
temp1=head;
printf("Employee Details \n");
while(temp1!=NULL)
{
printf(" \n");
printf("%s\n%s\n%s\n%s\n%d\n%lld\n",temp1->ssn,temp1->name,temp1->dept,temp1-
>designation,temp1->salary,temp1->phno);
printf(" ");
temp1=temp1->right;
}
}
void ins_beg(char s[15],char n[20],char dpt[5],char des[10],int sal,long long
int p)
{
temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
strcpy(temp1->ssn,s);
strcpy(temp1->name,n);
strcpy(temp1->dept,dpt);
strcpy(temp1->designation,des);
temp1->salary=sal;
temp1->phno=p;
temp1->right=head;
head->left=temp1;
head=temp1;
temp1->left=NULL;
}
```

```
void ins_end(char s[15],char n[20],char dpt[5],char des[10],int sal,long long
int p)
{
temp1=(struct Enode *)malloc(1*sizeof(struct Enode));
strcpy(temp1->ssn,s);
strcpy(temp1->name,n);
strcpy(temp1->dept,dpt);
strcpy(temp1->designation,des);
temp1->salary=sal;
temp1->phno=p;
tail->right=temp1;
temp1->left=tail;
temp1->right=NULL;
tail=temp1;
}
void del_beg()
{
temp1=head->right;
free(head);
head=temp1;
head->left=NULL;
}
void del_end()
{
temp1=tail->left;
free(tail);tail=temp1;
tail->right=NULL;
}
```

SAMPLE INPUT AND OUTPUT

```
-----------------MENU--------------------
1.Create 2.Display
3.Insert at beginning 4.Insert at End 5.Delete at beginning 6.Delete at End
7.Exit
```

```
------------------------------------------
Enter choice : 1
Enter no of employees : 2
Enter ssn,name,department, designation, salary and phno of employee : 1 RAJ
SALES MANAGER
15000 911
Enter ssn,name,department, designation, salary and phno of employee : 2 RAVI HR
ASST 10000
123
Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
No of employees = 2 Enter choice : 3
Enter ssn,name,department, designation, salary and phno of employee : 3 RAM
MARKET
MANAGER 50000 111
Enter choice : 2
Linked list elements from begining :
CSE DEPT,MSEC
Page 321 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
3 RAM MARKET MANAGER 50000.000000 111
No of employees = 3 Enter choice : 4
3 RAM MARKET MANAGER 50000.000000 111
Enter choice : 2
Linked list elements from begining :
1 RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
No of employees = 2
Enter choice : 5
Enter ssn,name,department, designation, salary and phno of employee :
0 ALEX EXE TRAINEE 2000 133
Enter choice : 2
Linked list elements from begining :
```

```
0
ALEX EXE TRAINEE 2000.000000 133
1
RAJ SALES MANAGER 15000.000000 911
2
RAVI HR ASST 10000.000000 123
No of employees = 3
Enter choice : 6
0
ALEX EXE TRAINEE 2000.000000 133
Enter choice : 2
Linked list elements from begining :1
RAJ SALES MANAGER 15000.000000 911
2 RAVI HR ASST 10000.000000 123
No of employees = 2
Enter choice : 7
Exit
```

PRACTICAL – 8

Design, Develop and Implement a Program for the following operationson Singly Circular Linked List (SCLL) with header nodes a.

Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
typedef struct poly_node
{
    float coef;
    int expx;
    int expy;
    int expz;
    struct poly_node *link;

}
POLY;
POLY *getNode();
void read_poly(POLY *head, int n);
void print_poly(POLY *head);
POLY *add_poly(POLY *h1, POLY *h2);
int compare(POLY *temp1, POLY *temp2);
void attach(float cf, POLY *exptemp, POLY **tempres);
POLY* delete(POLY *head, POLY *temp);
void evaluate(POLY *head);
void main()
{
    int n1, n2;
    POLY *POLY1 = getNode();
    POLY *POLY2 = getNode();
    POLY *POLYSUM = getNode();
POLY1->expx= -1;
POLY1->link= POLY1;
POLY2->link= POLY2;
POLYSUM->link= POLYSUM;
printf("\nEnter the number of terms for both polynomials\n");
scanf("%d%d",&n1, &n2);
printf("\nEnter 1st Polynomial\n");
read_poly(POLY1, n1);
printf("\n1st Polynomial is\n");
print_poly(POLY1);
printf("\nEnter 2nd Polynomial\n");
read_poly(POLY2, n2);
printf("\n2nd Polynomial is\n");
print_poly(POLY2);
POLYSUM = add_poly(POLY1, POLY2);
printf("\nThe Resultant polynomial is\n");
print_poly(POLYSUM);
evaluate(POLYSUM);

}

POLY *getNode()
{
    POLY *temp = (POLY *) malloc(sizeof(POLY));
    if(temp == NULL)
    {
        printf("No Memory\n");
        exit(0);
    }
    return temp;
```

```
}

void read_poly(POLY *head, int n)
{
    int i;
    POLY *new = NULL;
    POLY *temp = head;
    for(i=0; i<n; i++)
    {
        new = getNode();
        printf("Enter Coef and Exps\n");
        scanf("%f%d%d%d", &(new->coef), &(new->expx), &(new->expy), &(new-
>expz));
        (temp->link) = new;
        temp = temp->link;

    }
    temp->link= head;
    return;

}

void print_poly(POLY *head)
{
    POLY *temp = head->link;
    while(temp != head){
printf("%f*X^%d*Y^%d*Z^%d\t", temp->coef, temp->expx, temp->expy, temp->expz);
temp = temp->link;

    }
    printf("\n");
    return;

}
```

```
POLY *add_poly(POLY *h1, POLY *h2)
{
    float cf;
    POLY *temp1 = h1->link, *temp2 = NULL;
    POLY *result = getNode();
    POLY *tempres = result;
    while(temp1 != h1)
    {
        temp2 = h2->link;
        while(temp2 != h2)
        {
            switch(compare(temp1, temp2))
            {
                case 1:
                cf = temp1->coef+ temp2->coef;
                if(cf)
                {
                    attach(cf, temp1, &tempres);

                }
                temp1 = temp1->link;
                h2 = delete(h2, temp2);
                temp2 = h2->link;
                break;

                case 2:
                temp2 = temp2->link;
                break;

            }

        }
        if(temp1 != h1)
        {
            attach(temp1->coef, temp1, &tempres);
            temp1 = temp1->link;

        }

    }
    temp2 = h2->link;
    while(temp2 != h2)
    {
        attach(temp2->coef, temp2, &tempres);
        temp2 = temp2->link;

    }
    tempres->link= result;
    return result;

}

int compare(POLY *temp1, POLY *temp2)
{
if((temp1->expx== temp2->expx) && (temp1->expy== temp2->expy) && (temp1->expz==
temp2->expz))
{
    return 1;

}
return 2;

}

void attach(float cf, POLY *exptemp, POLY **tempres)
{
    POLY *new = getNode();
    new->coef= cf;
    new->expx= exptemp->expx;
    new->expy= exptemp->expy;
    new->expz= exptemp->expz;
    (*tempres)->link= new;
    *tempres = new;
    return ;

}

POLY* delete(POLY *head, POLY *temp)
{
    POLY *previous = head, *present = head->link;
    while(present != temp)
    {
        previous = present;
        present = present->link;

    }
previous->link= present->link;
free(present);
return head;

}

void evaluate(POLY *head)
{
    float result = 0.0;
    int x,y,z;
    POLY *temp = head->link;
    printf("\nEnter exponents\n");
    scanf("%d%d%d", &x, &y, &z);
    while(temp != head)
    {
        result += (temp->coef)*pow(x, temp->expx)*pow(y, temp->expy)*pow(z,
temp->expz);temp = temp->link;
```

```
    }
    printf("\nResult after evaluation is %f\n", result);
    return;

}
```

OUTPUT:

```
Enter the number of terms for both polynomials
2
2

Enter 1st Polynomial
Enter Coef and Exps
4
x
Enter Coef and Exps

1st Polynomial is
4.000000*X^0*Y^0*Z^0    0.000000*X^0*Y^0*Z^0

Enter 2nd Polynomial
Enter Coef and Exps
Enter Coef and Exps

2nd Polynomial is
0.000000*X^0*Y^0*Z^0    0.000000*X^0*Y^0*Z^0

The Resultant polynomial is
4.000000*X^0*Y^0*Z^0

Enter exponents

Result after evaluation is 4.000000
```

PRACTICAL – 9

Design, Develop and Implement a menu driven Program for the following operations on Binary Search Tree (BST) of Integers

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

c. Search the BST for a given element (KEY) and report the appropriate messagee.

Exit

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
struct BST
{
int data;
struct BST *left;
struct BST *right;
};
typedef struct BST NODE;
NODE *node;
NODE* createtree(NODE *node, int data)
{
if (node == NULL)
{
NODE *temp;
temp= (NODE*)malloc(sizeof(NODE));
temp->data = data;
temp->left = temp->right = NULL;
return temp;
```

```
}
if (data < (node->data))
{
node->left = createtree(node->left, data);
}
else if (data > node->data)
{
node -> right = createtree(node->right, data);
}
return node;
}
NODE* search(NODE *node, int data)
{
if(node == NULL)
printf("\nElement not found");
else if(data < node->data)
{
node->left=search(node->left, data);
}
else if(data > node->data)
{
node->right=search(node->right, data);
}
else
printf("\nElement found is: %d", node->data);
return node;
}
void inorder(NODE *node)
{
if(node != NULL)
{
inorder(node->left);
```

```
printf("%d\t", node->data);
inorder(node->right);
}
}
void preorder(NODE *node)
{
if(node != NULL)
{
printf("%d\t", node->data);
preorder(node->left);
preorder(node->right);
}
}
void postorder(NODE *node)
{
if(node != NULL)
{
postorder(node->left);
postorder(node->right);
printf("%d\t", node->data);
}
}
NODE* findMin(NODE *node)
{
if(node==NULL)
{
return NULL;
}
if(node->left)
return findMin(node->left);
else
return node;
```

```
}
NODE* del(NODE *node, int data)
{
NODE *temp;
if(node == NULL)
{
printf("\nElement not found");
}
else if(data < node->data)
{
node->left = del(node->left, data);
}
else if(data > node->data)
{
node->right = del(node->right, data);
}
else
{
if(node->right && node->left)
{
temp = findMin(node->right);
node -> data = temp->data;
node -> right = del(node->right,temp->data);
}
else
{
temp = node;
if(node->left == NULL)
node = node->right;
else if(node->right == NULL)
node = node->left;
free(temp);
```

```
}
}
return node;
}
void main()
{
int data, ch, i, n;
NODE *root=NULL;

while (1)
{
printf("\n1.Insertion in Binary Search Tree");
printf("\n2.Search Element in Binary Search Tree");
printf("\n3.Delete Element in Binary Search Tree");
printf("\n4.Inorder\n5.Preorder\n6.Postorder\n7.Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch (ch)
{
case 1:printf("\nEnter N value: " );
scanf("%d", &n);
printf("\nEnter the values to create BST like(6,9,5,2,8,15,24,14,7,8,5,2)\
n");
for(i=0; i<n; i++)
{
scanf("%d", &data);
root=createtree(root, data);
}
break;
case 2:printf("\nEnter the element to search: ");
scanf("%d", &data);
root=search(root, data);
break;
```

```
case 3:printf("\nEnter the element to delete: ");
scanf("%d", &data);
root=del(root, data);
break;
case 4:printf("\nInorder Traversal: \n");
inorder(root);
break;
case 5:printf("\nPreorder Traversal: \n");
preorder(root);
break;
case 6:printf("\nPostorder Traversal: \n");
postorder(root);
break;
case 7:exit(0);
default : printf("\nWrong option");
break;
}
}
}
```

SAMPLE INPUT AND OUTPUT

```
Program For Binary Search Tree
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :1 Enter The Element 15
Want To enter More Elements?(1/0)1 Enter The Element 25
Want To enter More Elements?(1/0)1 Enter The Element 35
Want To enter More Elements?(1/0)1
Enter The Element 45
Want To enter More Elements?(1/0)1 Enter The Element 5
Want To enter More Elements?(1/0)1 Enter The Element 7

Want To enter More Elements?(1/0)0 Enter your choice :2

Enter Element to be searched :7

The 7 Element is Present Parent of node 7 is 5

1.Create

2.Search

3.Recursive Traversals

4.Exit

Enter your choice :2

Enter Element to be searched :88

The 88 Element is not Present

Enter your choice :3

The Inorder display : 5 7 15 25 35 45

The Preorder display : 15 5 7 25 35 45

The Postorder display : 7 5 45 35 25 15

Enter your choice :4
```

PRACTICAL - 10

Design, Develop and Implement a Program for the following operations on Graph(G) of Cities
a.Create a Graph of N cities using AdjacencyMatrix.
b.Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
int a[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r=-1,count=0;
void bfs(int v)
{
for(i=1;i<=n;i++)
if(a[v][i] && !visited[i])
q[++r]=i;
if(f<=r)
{
visited[q[f]]=1;
bfs(q[f++]);
}
}
void dfs(int v)
{
int i; reach[v]=1;
for(i=1;i<=n;i++)
{
if(a[v][i] && !reach[i])
{
printf("\n %d->%d",v,i);
count++;
dfs(i);
}
}
}
void main()
{
int v, choice;
printf("\n Enter the number of vertices:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
q[i]=0;
visited[i]=0;
}
for(i=1;i<=n-1;i++)
reach[i]=0;
printf("\n Enter graph data in matrix form:\n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("1.BFS\n 2.DFS\n 3.Exit\n");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("\n Enter the starting vertex:");
scanf("%d",&v);
bfs(v);
if((v<1)||(v>n))
{
printf("\n Bfs is not possible");
}
else
{
printf("\n The nodes which are reachable from %d:\n",v);
for(i=1;i<=n;i++)
if(visited[i])
printf("%d\t",i);
}
break;
case 2:
dfs(1);
if(count==n-1)
printf("\n Graph is connected");
else
printf("\n Graph is not connected");
break;
case 3:
exit(0);
}
}
```

SAMPLE INPUT AND OUTPUT

```
Enter the number of vertices:5 Enter graph data in matrix form: 0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 0
0 1 0 0 0
1.
BFS
2.
DFS
3.
Exit 2
1->2
2->3
3->4
2->5
Graph is connectedcsdept Enter the number of vertices:5
Enter graph data in matrix form: 0 1 0 1 0 1 0 1 0 0
0 1 0 1 0
1 0 1 0 0
0 0 0 0 0
1.
2.
3.
BFS
DFS
Exit 2
1->2
2->3
```

```
3->4
Graph is not connected
Enter the number of vertices:5
Enter graph data in matrix form: 0 1 1 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
1.
BFS
2.
DFS
3.
Exit 1
Enter the starting vertex:1
The nodes which are reachable from 1: 2 3 4
Enter graph data in matrix form: 0 1 1 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
1.
BFS
2.
DFS
3.
Exit 1
Enter the starting vertex:0 BFS is not possible
```

PRACTICAL - 11

Write a Program to finds the position of an element in an array using Linear Search Algorithm and Binary search Algorithm.

PROGRAM :

Linear Search:

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
int a[20],n,x,i,flag=0;
cout<<"How many elements?";
cin>>n;
cout<<"\nEnter elements of the array\n";
for(i=0;i<n;++i)
cin>>a[i];
cout<<"\nEnter element to search:";
cin>>x;
for(i=0;i<n;++i)
{
if(a[i]==x)
{
flag=1;
break;
}
}
if(flag)
cout<<"\nElement is found at position "<<i+1;
else
cout<<"\nElement not found";
return 0;
}
```

OUTPUT

```
How many elements?5

Enter elements of the array
1 2 3 4 5

Enter element to search:5

Element is found at position 5
```

Binary Search:

```
#include <bits/stdc++.h>
```

```
using namespace std;

int binarySearch(int[], int, int, int);

int main()
{
   int num[10] = {10, 22, 37, 55, 92, 118};
   int search_num, loc=-1;

   cout<<"Enter the number that you want to search: ";
   cin>>search_num;

   loc = binarySearch(num, 0, 6, search_num);

   if(loc != -1)
   {
      cout<<search_num<<" found in the array at the location: "<<loc;
   }
   else
   {
      cout<<"Element not found";
   }
   return 0;
}

int binarySearch(int a[], int first, int last, int search_num)
{
   int middle;
   if(last >= first)
   {
      middle = (first + last)/2;
      if(a[middle] == search_num)
      {
         return middle+1;
      }

      else if(a[middle] < search_num)
      {
         return binarySearch(a,middle+1,last,search_num);
      }


      else
      {
         return binarySearch(a,first,middle-1,search_num);
      }

   }
   return -1;
}
```

OUTPUT:

```
Enter the number that you want to search: 92
92 found in the array at the location: 5
```

PRACTICAL - 12

Write a program to sort list using different sorting algorithms (bubble, selection, insertion, radix, merge and quick sort) and compare them.

PROGRAMS:

1. BUBBLE SORT

```
#include <bits/stdc++.h>
using namespace std;

void swap(int *xp, int *yp)
{
      int temp = *xp;
      *xp = *yp;
      *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
      int i, j;
      for (i = 0; i < n-1; i++)

      // Last i elements are already in place
      for (j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
                  swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
      int i;
      for (i = 0; i < size; i++)
            cout << arr[i] << " ";
      cout << endl;
}

// Driver code
int main()
{
      int arr[] = {64, 34, 25, 12, 22, 11, 90};
      int n = sizeof(arr)/sizeof(arr[0]);
      bubbleSort(arr, n);
      cout<<"Sorted array: \n";
      printArray(arr, n);
      return 0;
}
```

OUTPUT:

```
Sorted array:
11 12 22 25 34 64 90
```

2. SELECTION SORT

```
//SELECTION SORT
#include <iostream>
using namespace std;
int selection(int arr[],int);
void swap(int *xp, int *yp)
{
int temp = *xp;
*xp = *yp;
*yp = temp;
}
int main()
{
int arr[10]={2,4,1,0,6,9,8,3,5,7};
selection(arr,10);
for(int i=0;i<10;++i)
{
cout << arr[i];
}
return 0;
}
int selection(int arr[],int n)
{
int temp=0,i,j;
int min_index=0;
if(min_index<n-1)
{
for (i = 0; i < n-1; i++)
{
min_index = i;
for (j = i+1; j < n; j++)
{
if (arr[j] < arr[min_index])
min_index = j;
}
swap(&arr[min_index], &arr[i]);
}
}
return arr[n];
}
```

OUTPUT:

```
0123456789
```

3. Insertion Sort

```
#include<iostream>
using namespace std;
void display(int *array, int size) {
 for(int i = 0; i<size; i++)
   cout << array[i] << " ";
 cout << endl;
}
void insertionSort(int *array, int size) {
 int key, j;
 for(int i = 1; i<size; i++) {
   key = array[i];//take value
   j = i;
   while(j > 0 && array[j-1]>key) {
    array[j] = array[j-1];
    j--;
   }
   array[j] = key;  //insert in right place
}
}
int main() {
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   int arr[n];    //create an array with given number of elements
   cout << "Enter elements:" << endl;
   for(int i = 0; i<n; i++) {
      cin >> arr[i];
   }
   cout << "Array before Sorting: ";
   display(arr, n);
   insertionSort(arr, n);
   cout << "Array after Sorting: ";
   display(arr, n);
}
```

OUTPUT:

```
Enter the number of elements: 5
Enter elements:
5 4 3 2 1
Array before Sorting: 5 4 3 2 1
Array after Sorting: 1 2 3 4 5
```

4. Radiax Sort

```
#include<iostream>
#include<list>
```

```
#include<cmath>
using namespace std;
void display(int *array, int size) {
      for(int i = 0; i<size; i++)
       cout << array[i] << " ";
      cout << endl;
}
void radixSort(int *arr, int n, int max) {
      int i, j, m, p = 1, index, temp, count = 0;
      list<int> pocket[10];
      for(i = 0; i< max; i++) {
       m = pow(10, i+1);
       p = pow(10, i);
       for(j = 0; j<n; j++) {
      temp = arr[j]%m;
      index = temp/p;
      pocket[index].push_back(arr[j]);
       }
 count = 0;
 for(j = 0; j<10; j++) {
while(!pocket[j].empty()) {
 arr[count] = *(pocket[j].begin());
 pocket[j].erase(pocket[j].begin());
 count++;
}
 }
}
}
int main() {
   int n, max;
   cout << "Enter the number of elements: ";
   cin >> n;
   cout << "Enter the maximum digit of elements: ";
   cin >> max;
   int arr[n]; //create an array with given number of elements
   cout << "Enter elements:" << endl;
   for(int i = 0; i<n; i++) {
      cin >> arr[i];
   }
   cout << "Data before Sorting: ";
   display(arr, n);
   radixSort(arr, n, max);
   cout << "Data after Sorting: ";
   display(arr, n);
}
```

OUTPUT:

```
Enter the number of elements: 10
Enter the maximum digit of elements: 3
Enter elements:
802 630 20 745 52 300 612 932 78 187
Data before Sorting: 802 630 20 745 52 300 612 932 78 187
Data after Sorting: 20 52 78 187 300 612 630 745 802 932
```

5. MERGE SORT

```
// C++ program for Merge Sort
#include <bits/stdc++.h>
```

```
using namespace std;

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
      int n1 = m - l + 1;
      int n2 = r - m;

      // Create temp arrays
      int L[n1], R[n2];

      // Copy data to temp arrays L[] and R[]
      for (int i = 0; i < n1; i++)
            L[i] = arr[l + i];
      for (int j = 0; j < n2; j++)
            R[j] = arr[m + 1 + j];

      // Merge the temp arrays back into arr[l..r]

      // Initial index of first subarray
      int i = 0;

      // Initial index of second subarray
      int j = 0;

      // Initial index of merged subarray
      int k = l;

      while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                  arr[k] = L[i];
                  i++;
            }
            else {
                  arr[k] = R[j];
                  j++;
            }
            k++;
      }

      // Copy the remaining elements of
      // L[], if there are any
      while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
      }

      // Copy the remaining elements of
      // R[], if there are any
      while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
      }
}

// l is for left index and r is
// right index of the sub-array
// of arr to be sorted */
void mergeSort(int arr[],int l,int r){
      if(l>=r){
            return;//returns recursively
      }
      int m =l+ (r-l)/2;
      mergeSort(arr,l,m);
      mergeSort(arr,m+1,r);
      merge(arr,l,m,r);
}

// UTILITY FUNCTIONS
// Function to print an array
void printArray(int A[], int size)
{
      for (int i = 0; i < size; i++)
            cout << A[i] << " ";
}

// Driver code
int main()
{
      int arr[] = { 12, 11, 13, 5, 6, 7 };
      int arr_size = sizeof(arr) / sizeof(arr[0]);

      cout << "Given array is \n";
      printArray(arr, arr_size);

      mergeSort(arr, 0, arr_size - 1);

      cout << "\nSorted array is \n";
      printArray(arr, arr_size);
      return 0;
}
```

OUTPUT:

```
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13
```

6. QUICK SORT

```
/* C++ implementation of QuickSort */
#include <bits/stdc++.h>
using namespace std;
```

```c
// A utility function to swap two elements
void swap(int* a, int* b)
{
	int t = *a;
	*a = *b;
	*b = t;
}

/* This function takes last element as pivot, places
the pivot element at its correct position in sorted
array, and places all smaller (smaller than pivot)
to left of pivot and all greater elements to right
of pivot */
int partition (int arr[], int low, int high)
{
	int pivot = arr[high]; // pivot
	int i = (low - 1); // Index of smaller element and indicates the
right position of pivot found so far

	for (int j = low; j <= high - 1; j++)
	{
		// If current element is smaller than the pivot
		if (arr[j] < pivot)
		{
			i++; // increment index of smaller element
			swap(&arr[i], &arr[j]);
		}
	}
	swap(&arr[i + 1], &arr[high]);
	return (i + 1);
}

/* The main function that implements QuickSort
arr[] --> Array to be sorted,
low --> Starting index,
high --> Ending index */
void quickSort(int arr[], int low, int high)
{
	if (low < high)
	{
		/* pi is partitioning index, arr[p] is now
		at right place */
		int pi = partition(arr, low, high);

		// Separately sort elements before
		// partition and after partition
		quickSort(arr, low, pi - 1);
		quickSort(arr, pi + 1, high);
	}
}

/* Function to print an array */
void printArray(int arr[], int size)
{
	int i;
	for (i = 0; i < size; i++)
		cout << arr[i] << " ";
	cout << endl;
}

// Driver Code
int main()
{
	int arr[] = {10, 7, 8, 9, 1, 5};
	int n = sizeof(arr) / sizeof(arr[0]);
	quickSort(arr, 0, n - 1);
	cout << "Sorted array: \n";
	printArray(arr, n);
	return 0;
}
```

OUTPUT:

```
Sorted array:
1 5 7 8 9 10
```

COMPARISON OF SORTING ALGORITHMS

Sorting Algorithms	**Best Case (Time Complexity)**	**Average Case (Time Complexity)**	**Worst Case (Time Complexity)**	**Worst Case (Space complexity)**
Bubble Sort	$\Omega(N)$	$\theta(N^2)$	$O(N^2)$	$O(1)$
Selection Sort	$\Omega(N^2)$	$\theta(N^2)$	$O(N^2)$	$O(1)$
Insertion Sort	$\Omega(N)$	$\theta(N^2)$	$O(N^2)$	$O(1)$
Radix Sort	$\Omega(N\ k)$	$\theta(N\ k)$	$O(N\ k)$	$O(N + k)$
Merge Sort	$\Omega(N \log N)$	$\theta(N \log N)$	$O(N \log N)$	$O(N)$
Quick Sort	$\Omega(N \log N)$	$\theta(N \log N)$	$O(N^2)$	$O(N \log N)$