Data Structure Lab with C and C++
Subject Code: MCAL11

A Practical Journal Submitted in Fulfillment
of the Degree of

MASTER
In
COMPUTER APPLICATION
Year 2022-2023

By
(Ravishankar Jaiswal)
(172047)
Semester- 1
Under the Guidance of

MS. Kavita Chouk

Institute of Distance and Open Learning
Vidya Nagari, Kalina, Santacruz East – 400098.
University of Mumbai

PCP Center
[Satish Pradhan Dyanasadhana College, Thane]

Institute of Distance and Open Learning,

Vidyanagari, Kalina, Santacruz (E) -400098

CERTIFICATE

This to certify that, **(Ravishankar Jaiswal)** appearing **Master in Computer Application (Semester I) Application ID: 172047**has satisfactory completed the prescribed practical of **MCAL11- Data structure Lab with C and C++** as laid down by the University of Mumbai for the academic year 2022-23

Teacher in charge

Examiners

Coordinator
IDOL, MCA
University of Mumbai

Date: -01/04/2023
Place: - THANE

Practical 1

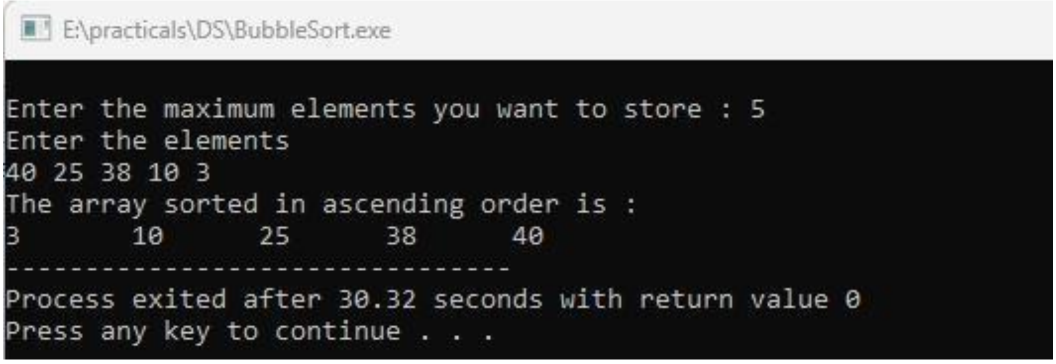
1. Bubble Sort

Aim: Implement program for Bubble Sort

Program:

```
#include  <stdio.h>
#include <stdlib.h>
int main()
{
 int i, n, temp, j, arr[10];
 printf("\nEnter the maximum elements you want to store : ");
 scanf("%d", &n);
 printf("Enter the elements \n");
 for(i=0;i<n;i++)
  {
 scanf("%d", & arr[i]);
  }
  for(i=0;i<n;i++)
  {
 for(j=0;j<n-1;j++)
  {
 if(arr[j]>arr[j+1])
  {
 temp = arr[j];
 arr[j] = arr[j+1];
 arr[j+1] = temp;
  }
  }
  }
 printf("The array sorted in ascending order is :\n");
 for(i=0;i<n;i++)
 printf("%d\t", arr[i]);
 return 0;
}
```

Output:



```
Enter the maximum elements you want to store : 5
Enter the elements
40 25 38 10 3
The array sorted in ascending order is :
3       10      25      38      40
--------------------------------
Process exited after 30.32 seconds with return value 0
Press any key to continue . . .
```

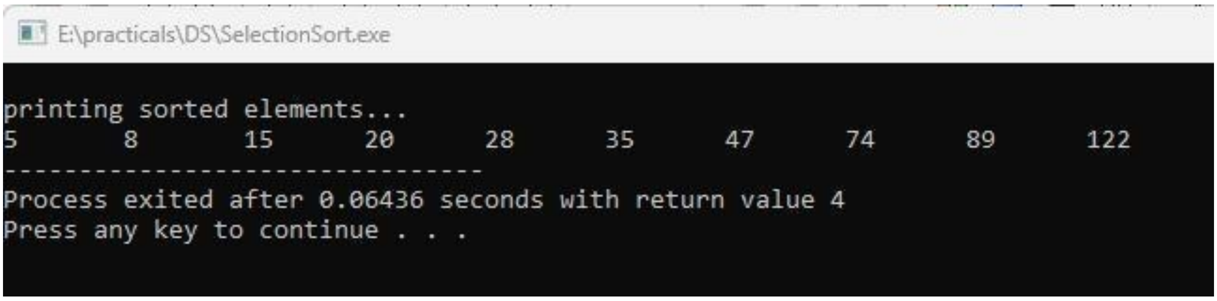
2. Selection Sort

Aim: Implement program for Selection Sort

Program:

```
#include<stdio.h>
#include<conio.h>
int smallest(int[],int,int);
void main ()
{
 int a[10] = {15,8,20,35,28,47,122,74,89,5};
 int i,j,k,pos,temp;
 for(i=0;i<10;i++)
 {
 pos = smallest(a,10,i);
 temp = a[i];
 a[i]=a[pos];
 a[pos] = temp;
 }
 printf("\nprinting sorted elements...\n");
 for(i=0;i<10;i++)
 {
 printf("%d\n",a[i]);
 }
}
int smallest(int a[], int n, int i)
{
 int small,pos,j;
 small = a[i];
 pos = i;
 for(j=i+1;j<10;j++)
 {
 if(a[j]<small)
 {
 small = a[j];
 pos=j;
 }
 }
 return pos;
}
```

Ouput:



```
printing sorted elements...
5       8       15      20      28      35      47      74      89      122
--------------------------------
Process exited after 0.06436 seconds with return value 4
Press any key to continue . . .
```

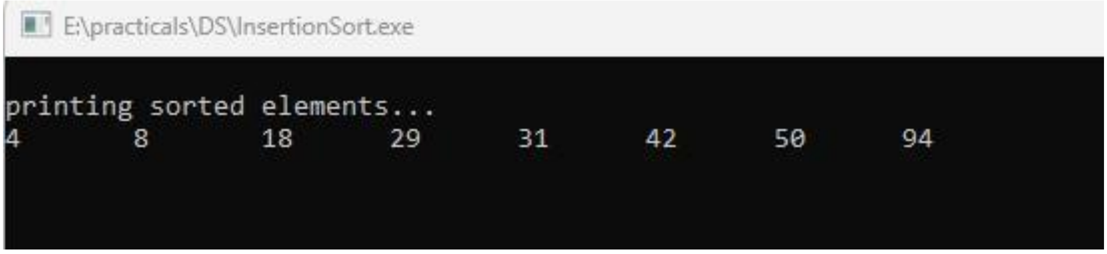
3. Insertion Sort

Aim:Implement program for Insertion Sort

Program:

```
#include<stdio.h>
#include<conio.h>
int main ()
{
 int i, j, k,temp;
 int a[8] = {18,29,4,8,42,31,94,50};
 printf("\nprinting sorted elements...\n");
 for(k=1; k<8; k++)
 {
 temp = a[k];
 j= k-1;
 while(j>=0 && temp <= a[j])
 {
 a[j+1] = a[j];
 j = j-1;
 }
 a[j+1] = temp;
 }
 for(i=0;i<8;i++)
 {
 printf("%d\t",a[i]);
 }
getch();
return 0;
}
```

Output:



```

printing sorted elements...
4       8       18      29      31      42      50      94
```

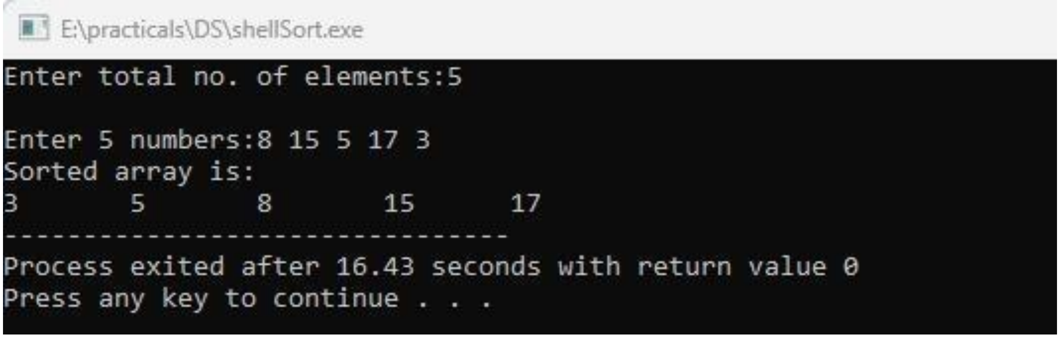
4. Shell

Aim:Implement program for Shell Sort

Program:

```
#include<stdio.h>
#include<conio.h>
int shellsort(int arr[], int num)
{
int i,j,k,tmp;
for(i=num/2;i>0;i=i/2)
{
for(j=i;j<num;j++)
{
for(k=j-i;k>=0;k=k-i)
{
if(arr[k+i]>=arr[k])
break;
else
{
tmp=arr[k];
arr[k]=arr[k+i];
arr[k+i]=tmp;
}
}
}}}
int main()
{
int arr[30];
int k,num;
printf("Enter total no. of elements:");
scanf("%d",&num);
printf("\nEnter %d numbers:",num);
for(k=0;k<num;k++)
{
scanf("%d",&arr[k]);
}
shellsort(arr,num);
printf("Sorted array is:\n");
for(k=0;k<num;k++)
printf("%d\t",arr[k]);
return 0;
}
```

Output:



```
Enter total no. of elements:5

Enter 5 numbers:8 15 5 17 3
Sorted array is:
3       5       8       15      17
--------------------------------
Process exited after 16.43 seconds with return value 0
Press any key to continue . . .
```

Practical 2

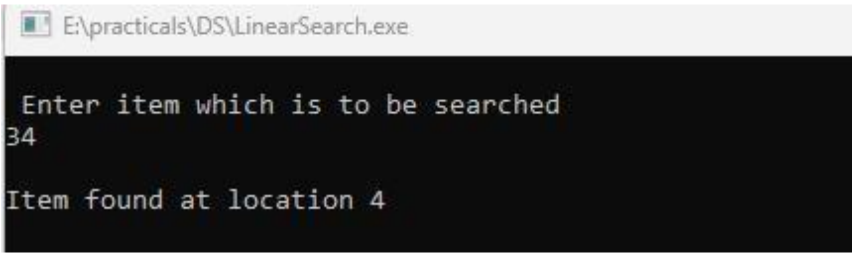
1. Linear Search:

Aim: Implement program for Linear Search

Program:

```
#include<stdio.h>
#include<conio.h>
int main()
{
int a[10]={15,25,4,34,47,8,112,73,5,67};
int item,i,flag;
printf("\n Enter item which is to be searched\n");
scanf("%d",&item);
for(i=0;i<10;i++)
{
if(a[i]==item)
{
flag=i+1;
break;
}
else
{
flag=0;
}
}
if(flag!=0)
{
printf("\nItem found at location %d\n",flag);
}
else
{
printf("\nItem not found\n");
}
getch();
}
```

Output:



E:\practicals\DS\LinearSearch.exe

```
 Enter item which is to be searched
34

Item found at location 4
```

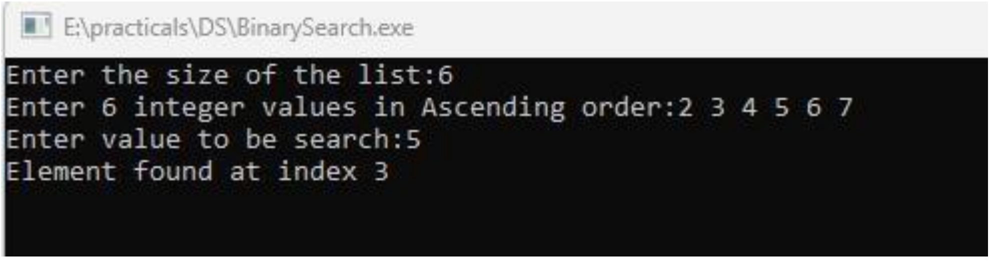
2. Binary Search

Aim: Implement program for Binary Search

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int f,l,m,s,i,k,list[100];
printf("Enter the size of the list:");
scanf("%d",&s);
printf("Enter %d integer values in Ascending order:",s);
for(i=0;i<s;i++)
{
scanf("%d",&list[i]);
}
printf("Enter value to be search:");
scanf("%d",&k);
f=0;
l=s-1;
m=(f+l)/2;
while(f<=l)
{
if(list[m]<k)
{
f=m+1;
}
else if(list[m]==k)
{
printf("Element found at index %d \n",m);
break;
}
else
{
l=m-1;}
m=(f+l)/2;
}
if(f>l)
{
printf("Element Not found in the list");
}
getch();
}
```

Output:



```
Enter the size of the list:6
Enter 6 integer values in Ascending order:2 3 4 5 6 7
Enter value to be search:5
Element found at index 3
```

Practical 3

1. Stack Using Array

Aim: Implement program for Stack using array:

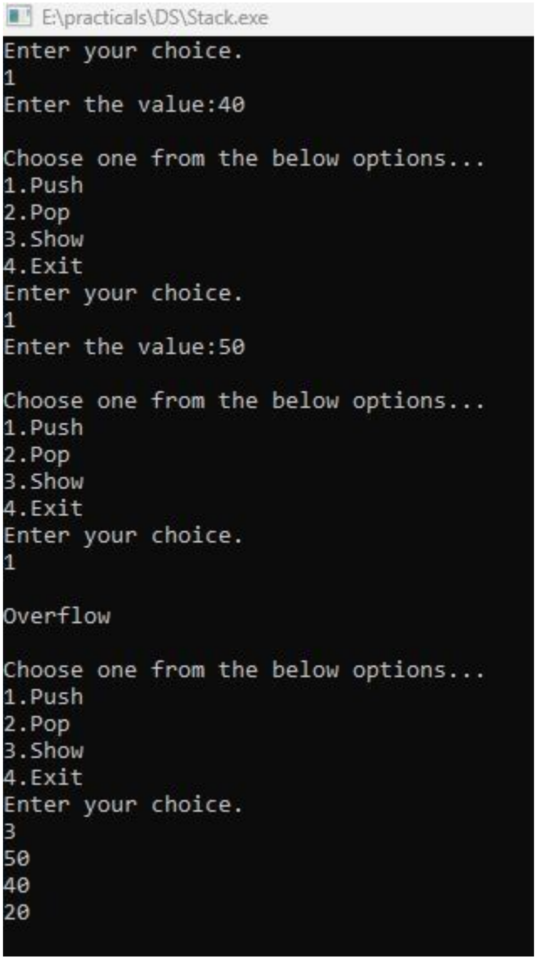
Program:

```
#include<stdio.h>
#include<conio.h>
int stack[10],i,j,choice=0,n,top=-1;
void push();
void pop();
void show();
int main()
{
  printf("Enter the number of elements in the stack:");
  scanf("%d",&n);
  while(choice!=4)
  {
  printf("\nChoose one from the below options...\n");
  printf("1.Push\n2.Pop\n3.Show\n4.Exit");
  printf("\nEnter your choice.\n");
  scanf("%d",&choice);
  switch(choice)
  {
              case 1:
              {
                          push();
                          break;
              }
              case 2:
              {
                          pop();
                          break;
              }
              case 3:
              {
                          show();
                          break;
              }
              case 4:
              {
                          printf("Existing...");
                          break;
              }
              default:
              {
                          printf("Please Enter valid choice:");
              }
  }
  }
}

void push()
{
  int val;
  if(top==n)
  printf("\nOverflow\n");
```

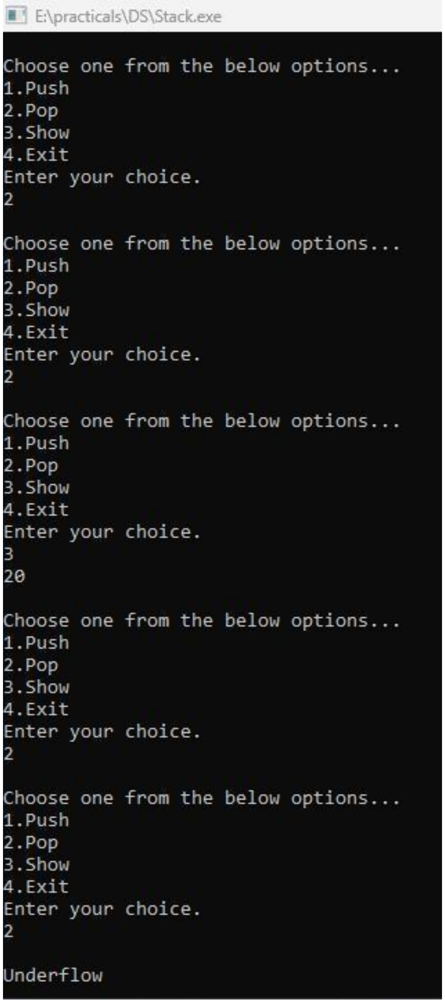
else

```
  {
   printf("Enter the value:");
   scanf("%d",&val);
   top=top+1;
   stack[top]=val;
  }
}
void pop()
{
  if(top==-1)
  printf("\nUnderflow\n");
  else
  top=top-1;
}
void show()
{
  for(i=top;i>=0;i--)
  {
            printf("%d\n",stack[i]);
  }
  if(top==-1)
  {
            printf("Stack is empty.");
  }
}
```

Output:



```
Enter your choice.
1
Enter the value:40

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
1
Enter the value:50

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
1

Overflow

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
3
50
40
20
```



```
Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
2

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
2

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
3
20

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
2

Choose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice.
2

Underflow
```

2. LinkedList:

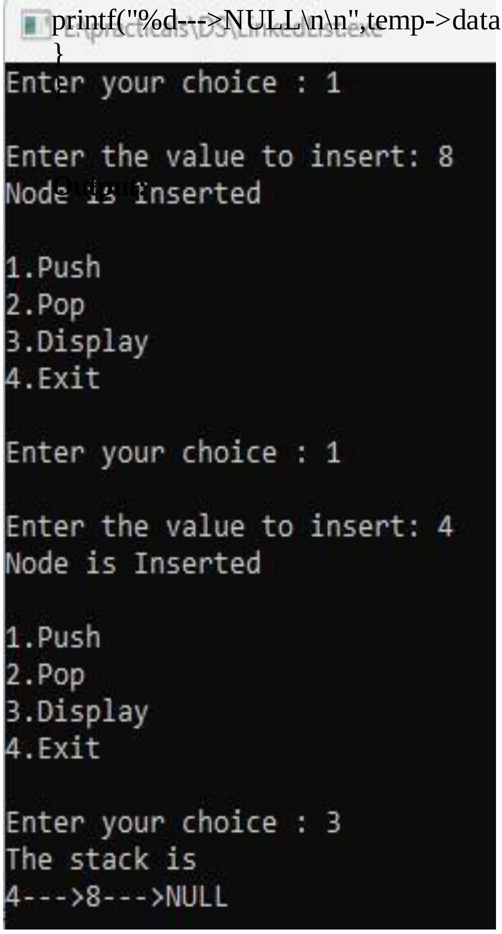
Aim: Implement program for stack using LinkedList

Program:

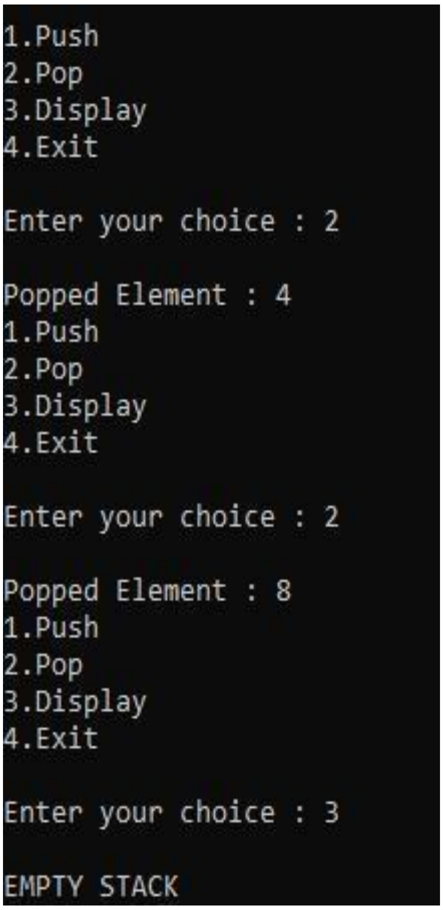
```
#include<stdio.h>
#include<stdlib.h>
/* Structure to create a node with data and pointer */
struct Node
{
int data;
struct Node *next;
}
*top = NULL; // Initially the list is empty
void push(int);
void pop();
void display();
int main()
{
 int choice, value;
 printf("Implementing Stack Using Linked List\n");
while(1)
{
printf("1.Push\n2.Pop\n3.Display\n4.Exit\n");
printf("\nEnter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\nEnter the value to insert: ");
scanf("%d", &value);
push(value);
break;
case 2: pop();
break;
case 3: display();
break;
case 4: exit(0);
break;
default: printf("\nInvalid Choice\n");
}
}
}
void push(int value)
{
struct Node *newNode;
newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value; // get value for the node
if(top == NULL)
newNode->next = NULL;
else
newNode->next = top; // Make the node as TOP
top = newNode;
printf("Node is Inserted\n\n");
}
```

```
void pop()
{
if(top == NULL)
printf("\nEMPTY STACK\n");

else{

struct Node *temp = top;
printf("\nPopped Element : %d", temp->data);
printf("\n");
top = temp->next; // After popping, make the next node as TOP
free(temp);
}
}
void display()
{
if(top == NULL)
printf("\nEMPTY STACK\n");
else
{
printf("The stack is \n");
struct Node *temp = top;
while(temp->next != NULL){
printf("%d--->",temp->data);
temp = temp -> next;
}
printf("%d--->NULL\n\n",temp->data);
}
}
```

Output:



```
Enter your choice : 1

Enter the value to insert: 8
Node is inserted

1.Push
2.Pop
3.Display
4.Exit

Enter your choice : 1

Enter the value to insert: 4
Node is Inserted

1.Push
2.Pop
3.Display
4.Exit

Enter your choice : 3
The stack is
4--->8--->NULL
```



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

Enter your choice : 2

Popped Element : 4
1.Push
2.Pop
3.Display
4.Exit

Enter your choice : 2

Popped Element : 8
1.Push
2.Pop
3.Display
4.Exit

Enter your choice : 3

EMPTY STACK
```

Practical 4

1. Linked List

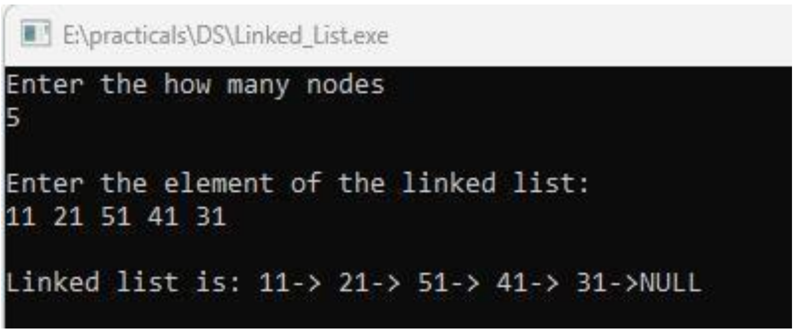
Aim:Implement program for Linked List.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node* create(int n);
void display(struct node* head);
void main()
{
int n=0;
//clrscr();
struct node* head=NULL;
printf("Enter the how many nodes\n");
scanf("%d",&n);
head=create(n);
display(head);
getch();
}

struct node* create(int n)
{
int i=0;
struct node* head=NULL;
struct node* temp=NULL;
struct node* p=NULL;
printf ("\nEnter the element of the linked list:\n");
for(i=0;i<n;i++)
{
temp=(struct node*)malloc(sizeof(struct node*));
scanf("%d",&temp->data);
temp->next=NULL;
if(head==NULL)
{
head=temp;
}
else
{
p=head;
while(p->next!=NULL)
p=p->next;
p->next=temp;
}
}
return head;
}
```

```
void display(struct node*head)
{
printf("\nLinked list is:");
struct node* p=head;
while(p!=NULL)
{
printf(" %d",p->data);
printf("->");
p=p->next;
}
printf("NULL");
}
```

Output:



E:\practicals\DS\Linked_List.exe

```
Enter the how many nodes
5

Enter the element of the linked list:
11 21 51 41 31

Linked list is: 11-> 21-> 51-> 41-> 31->NULL
```

Practical 5

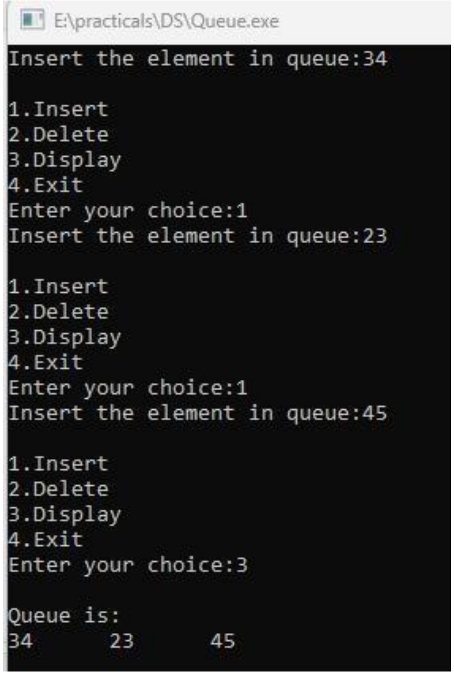
1. Queue

Aim:Implement program for Queue.

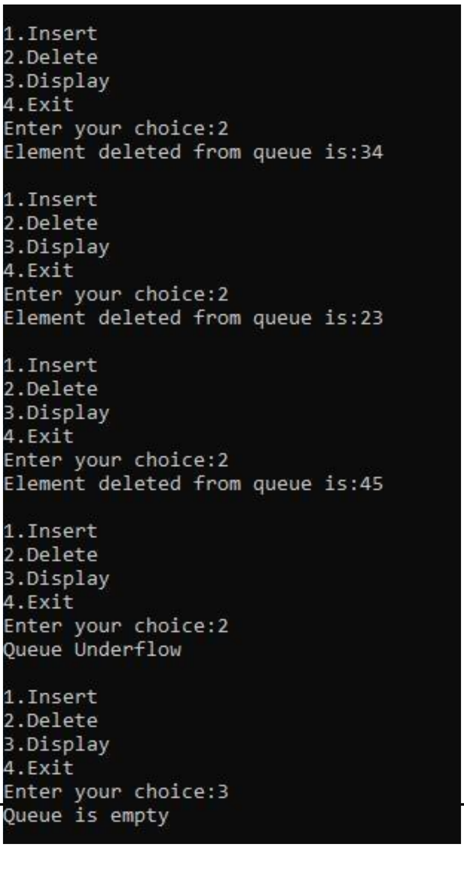
Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 50
int queue_array[MAX];
int rear = -1;
int front = -1;
insert()
{
int add_item;
if(rear==MAX-1)
printf("Queue Overflow\n");
else
{
if(front == -1)
front=0;
printf("Insert the element in queue:");
scanf("%d",&add_item);
rear=rear+1;
queue_array[rear]=add_item;
}
return 1;
}

deleteq()
{
if(front == -1 || front>rear)
{
printf("Queue Underflow\n");
return 1;
}
else
{
printf("Element deleted from queue is:");
printf("%d",queue_array[front]);
printf("\n");
front=front+1;
}
return 1;
}
display()
{
int i;

if(front == -1 || front>rear)
{
printf("Queue is empty\n");
}
else
{
printf("\nQueue is:\n");
for(i=front;i<=rear;i++)
printf("%d\t",queue_array[i]);
printf("\n");
}
return 1;
}

void main()
{
int ch;
while(1)
{
printf("\n");
printf("1.Insert\n");
printf("2.Delete\n");
printf("3.Display\n");
printf("4.Exit\n");
printf("Enter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1: insert();
 break;
case 2: deleteq();
 break;
case 3: display();
 break;
case 4: exit(0);
break;
default: printf("\n Wrong choice\n");
}
}
}
```

Output:



```
Insert the element in queue:34

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Insert the element in queue:23

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Insert the element in queue:45

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

Queue is:
34      23      45
```



```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Element deleted from queue is:34

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Element deleted from queue is:23

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Element deleted from queue is:45

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

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

Practical 6

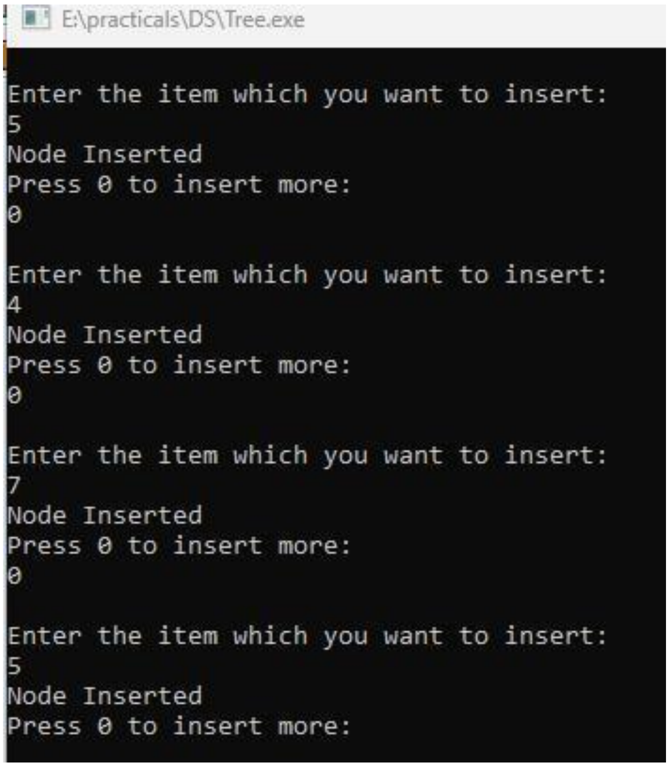
1. Binary Search Tree

Aim: Creating Binary Search Tree

Program:

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

void insert(int);
struct node
{
int data;
struct node *left;
struct node *right;
};
struct node *root;
int main ()
{
int choice,item;
do
{
printf("\nEnter the item which you want to insert:\n");
scanf("%d",&item);
insert(item);
printf("Press 0 to insert more:\n");
scanf("%d",&choice);
}while(choice == 0);
return 0;
}
void insert(int item)
{
struct node *ptr, *parentptr , *nodeptr;
ptr = (struct node *) malloc(sizeof (struct node));
if(ptr == NULL)
{
printf("cannot insert");
}
else
{
ptr -> data = item;
ptr -> left = NULL;
ptr -> right = NULL;
if(root == NULL)
{
root = ptr;
root -> left = NULL;
root -> right = NULL;
}
else
{
parentptr = NULL;
nodeptr = root;
while(nodeptr != NULL)
{
parentptr = nodeptr;
if(item < nodeptr->data)
{
nodeptr = nodeptr -> left;
}
else
{
nodeptr = nodeptr -> right;
}
}
if(item < parentptr -> data)
{
parentptr -> left = ptr;
}
else
{
parentptr -> right = ptr;
}
}
printf("Node Inserted\n");
}
}
```

Output:



E:\practicals\DS\Tree.exe

```

Enter the item which you want to insert:
5
Node Inserted
Press 0 to insert more:
0

Enter the item which you want to insert:
4
Node Inserted
Press 0 to insert more:
0

Enter the item which you want to insert:
7
Node Inserted
Press 0 to insert more:
0

Enter the item which you want to insert:
5
Node Inserted
Press 0 to insert more:
```

Practical 7

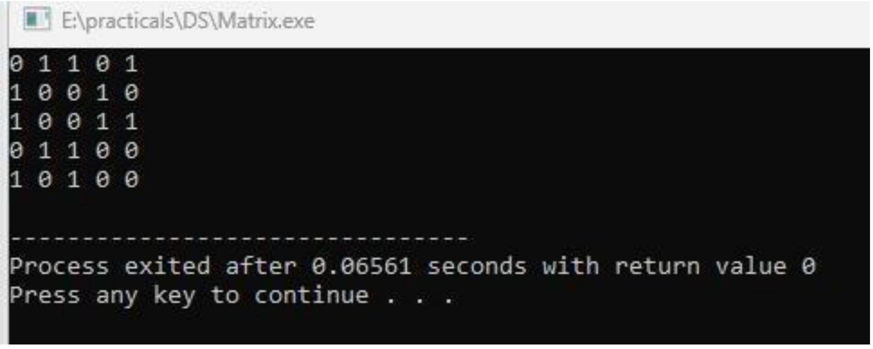
1. Adjacent Matrix of graph

Aim: Graph creation using adjacency matrix

Program:

```
#include<stdio.h>
int vertArr[20][20]; //the adjacency matrix initially 0
int count = 0;
void displayMatrix(int v) {
int i, j;
for(i = 0; i < v; i++) {
for(j = 0; j < v; j++) {
printf("%d ",vertArr[i][j]);
}
printf("\n");
}
}
void add_edge(int u, int v) { //function to add edge into the matrix
vertArr[u][v] = 1;
vertArr[v][u] = 1;
}
int main() {
int v = 5; //there are 6 vertices in the graph
add_edge(0, 1);
add_edge(0, 2);
add_edge(0, 4);
add_edge(1, 3);
add_edge(3, 2);
add_edge(2, 4);
displayMatrix(v);
return 0;
}
```

Output:



```
0 1 1 0 1
1 0 0 1 0
1 0 0 1 1
0 1 1 0 0
1 0 1 0 0

--------------------------------
Process exited after 0.06561 seconds with return value 0
Press any key to continue . . .
```

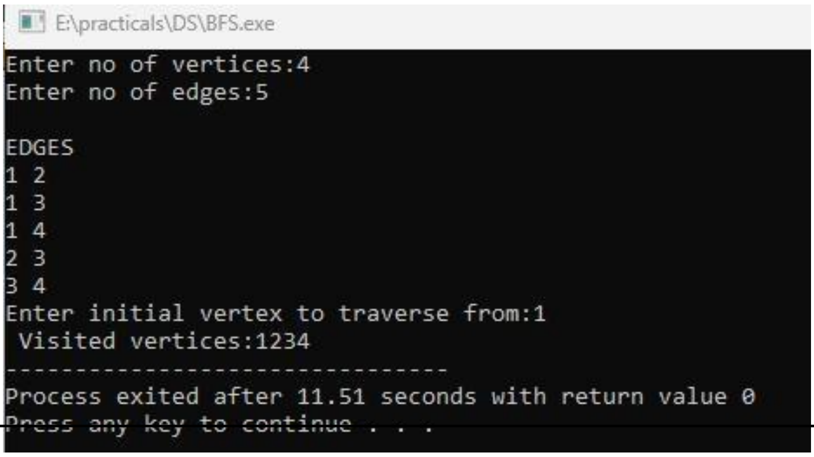
2. Print BFS:

Aim: Performing BFS traversal on Graph Data Structure

Program:

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

int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10];
int main()
{
int m;
printf("Enter no of vertices:");
scanf("%d",&n);
printf("Enter no of edges:");
scanf("%d",&m);
printf("\nEDGES \n");
for(k=1; k<=m; k++)
{
scanf("%d %d",&i,&j);
cost[i][j]=1;
}
printf("Enter initial vertex to traverse from:");
scanf("%d",&v);
printf(" ");
printf("Visited vertices:");
printf("%d",v);
visited[v]=1;
k=1;
while(k<n)
{
for(j=1; j<=n; j++)
if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
{
visit[j]=1;
qu[rare++]=j;
}
v=qu[front++];
printf("%d",v);
k++;
visit[v]=0;
visited[v]=1;
}
return 0;
}
```

Output:



E:\practicals\DS\BFS.exe

```
Enter no of vertices:4
Enter no of edges:5

EDGES
1 2
1 3
1 4
2 3
3 4
Enter initial vertex to traverse from:1
 Visited vertices:1234
--------------------------------
Process exited after 11.51 seconds with return value 0
Press any key to continue . . .
```