**EXPERIMENT NO 1**

**DATE: 08/09/2021**

**Stack Implementation using Array**

**AIM**: Implementing stack using array concept.

**PROGRAM:**

#include<stdio.h>

int stack[20];

int option,n,top,a,i;

void push();

void pop();

void display();

int main()

{

top = -1;

printf("B\_2060425\_BILHA P ABY \n");

printf("\n Enter the size of STACK : ");

scanf("%d",&n);

printf(" ---------------------------------------------\n");

printf("|STACK IMPLEMENTATION USING ARRAYS |\n");

printf(" ---------------------------------------------\n");

do

{

printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n");

printf("\nEnter the option no. : ");

scanf("%d",&option);

switch(option)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

break;

}

default:

{printf ("\nInvalid option\n");

}}}

while(option!=4);

return 0;

}

void push()

{

if(top >= n - 1)

{

printf("\nSTACK OVERFLOW\n");

}

else

{

printf("Enter a value to be pushed : ");

scanf("%d",&a);

top++;

stack[top] = a;

}}

void pop()

{

if(top <= -1)

{

printf("\nSTACK UNDERFLOW\n");

}

else

{

printf("\nThe popped element is %d",stack[top]);

top--;

}}void display()

{

if(top >= 0)

{

printf("\nELEMENTS IN THE STACK\n\n");

for(i = top ; i >= 0 ; i--)

printf("%d\t",stack[i]);

}

else

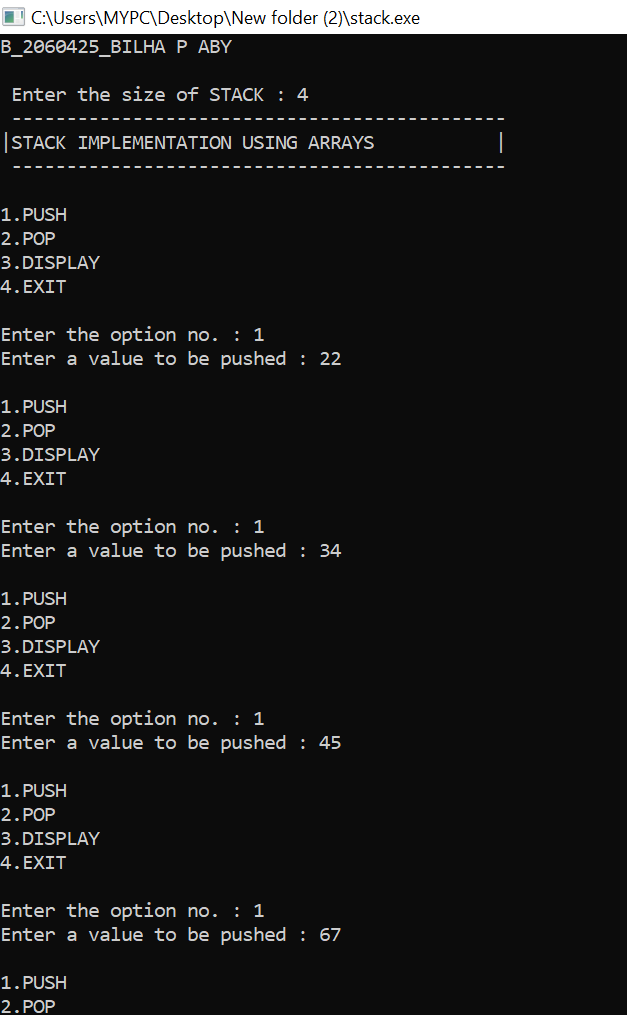
{

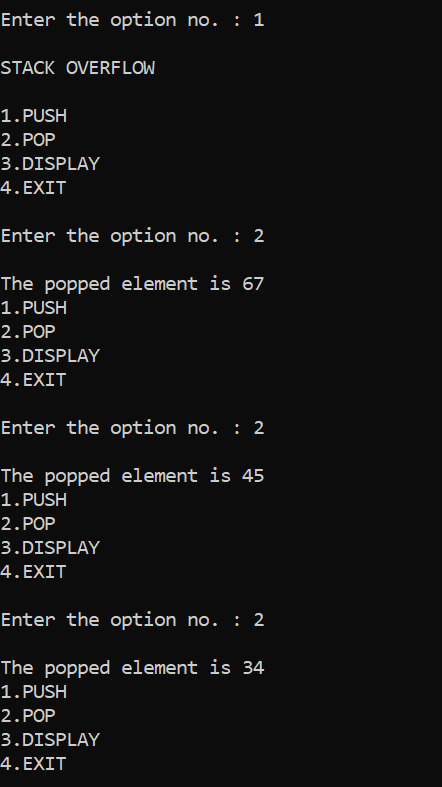
printf("\nEMPTY STACK\n");

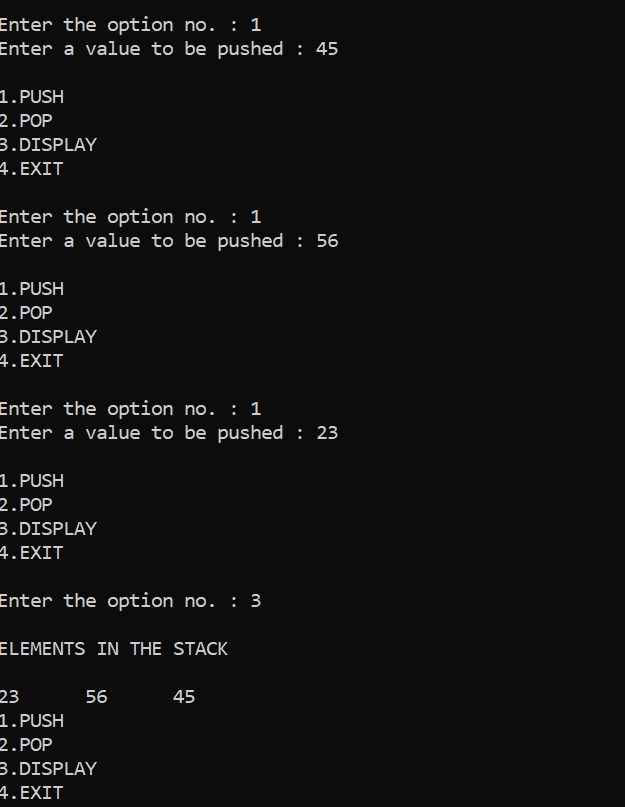
}

}

**OUTPUT:**







**RESULT:** Stack is implemented using arrays.

**EXPERIMENT NO 2**

**DATE: 22/09/2021**

**Queue Implementation using Array**

**AIM**: Implementing Queue using array concept.

**PROGRAM:**

#include<stdio.h>

int queue[20];

int front=-1;

int rear =-1;

int n,option;

void enqueue();

void dequeue();

void display();

int main()

{

printf("B\_2060425\_BILHA P ABY\n");

printf("\n Enter the size of Queue : ");

scanf("%d",&n);

printf(" ---------------------------------------------\n");

printf("|QUEUE IMPLEMENTATION USING ARRAYS |\n");

printf(" ---------------------------------------------\n");

do

{

printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.EXIT\n");

printf("\nEnter the option no. : ");

scanf("%d",&option);

switch(option)

{

case 1:

{

enqueue();

break;

}

case 2:

{

dequeue();

break;

}

case 3:

{

display();

break;

}

case 4:

{

break;

}

default:

{

printf ("\nInvalid option\n");

}}}

while(option!=4);

return 0;

}

void enqueue()

{

if(rear == n - 1)

{

printf("\n-------QUEUE IS FULL-------");

}

else{

if(front == -1) front = 0;

rear++;

printf("Element to be enqueued: ");

scanf("%d", &queue[rear]);

}}

void dequeue()

{

if(front==-1 && rear==-1)

{

printf("\_\_\_\_\_NO MORE ELEMENT TO DEQUEUE\_\_\_\_\_");

}

else{

printf("The Dequeued element from the queue is %d", queue[front]);

front++;

if(front>rear)

{

front=-1;

rear=-1;

}

}

}void display()

{ int i;

if(front==-1 && rear==-1)

{

printf("------The Queue is EMPTY------");

}

else{

for(i=front;i<rear+1;i++)

{

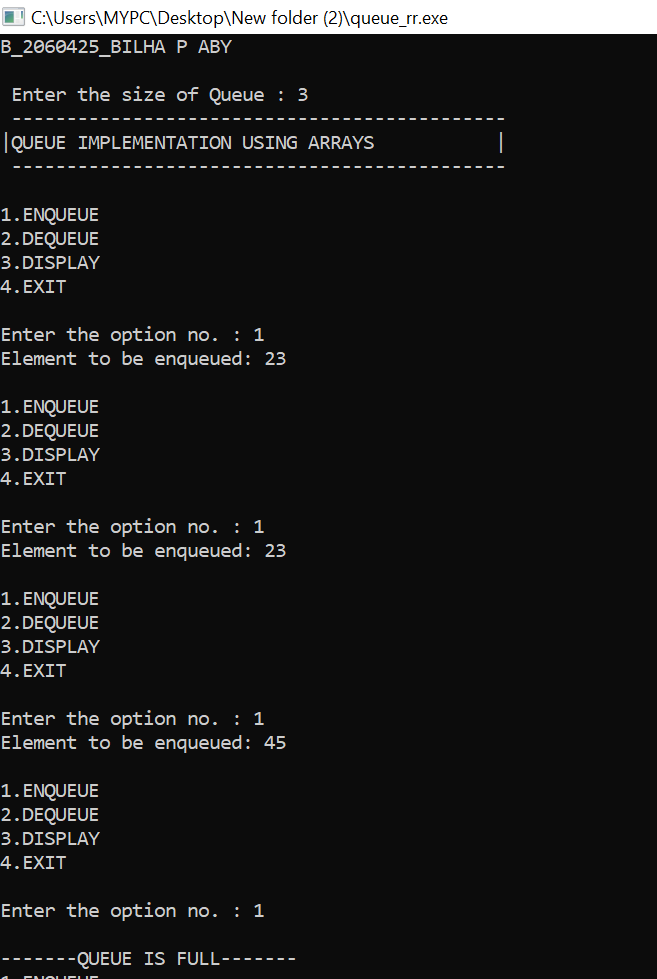
printf("\t%d\t|", queue[i]);

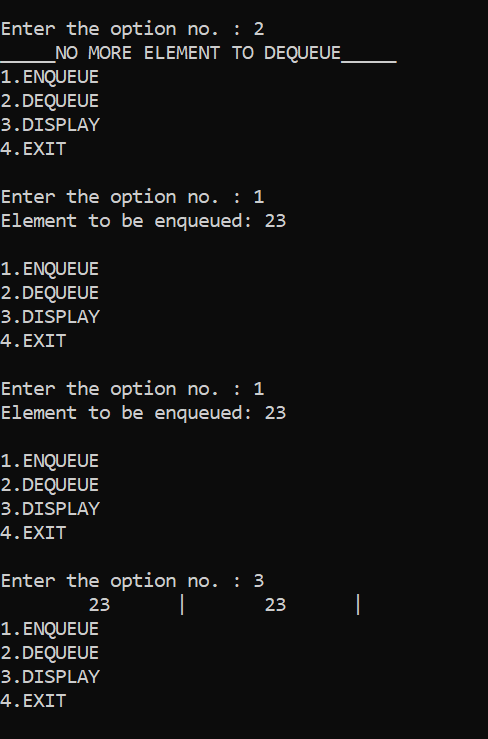
}

}

}

**OUTPUT:**





**RESULT:** Queue is implemented using arrays.

**EXPERIMENT NO 3**

**DATE: 29/09/2021**

**Singly Linked List**

**AIM**: Implementing of singly Linked list.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

void insert\_begin(int data)

{

struct node \*newnode;

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=data;

if(head==NULL)

{

newnode->next=NULL;

head=newnode;

}

else

{

newnode->next=head;

head=newnode;

}

}

void insert\_end(int data)

{

struct node \*newnode;

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=data;

if(head==NULL)

{

newnode->next=NULL;

head=newnode;

}

else

{

struct node \*temp=head;

while(temp->next!=NULL)

temp=temp->next;

temp->next=newnode;

newnode->next=NULL;

}

}

void delete\_begin()

{

if(head==NULL)

printf("\n\_\_\_\_\_\_\_LIST IS EMPTY\_\_\_\_\_\_\n");

else

head=head->next;

}

void delete\_end()

{

if(head==NULL)

printf("\n\_\_\_\_\_\_\_LIST IS EMPTY\_\_\_\_\_\_\n");

else if(head->next==NULL)

head=NULL;

else

{

struct node \*temp=head;

while(temp->next->next!=NULL)

{

temp=temp->next;

}

temp->next=NULL;

}

}

void display()

{

struct node \*temp=head;

if(head==NULL)

printf("\n\_\_\_\_\_\_\_LIST IS EMPTY\_\_\_\_\_\_\n");

else

{

while(temp!=NULL)

{

printf("%d -> ",temp->data);

temp=temp->next;

}

printf(" NULL");

}

}

int main()

{

int i,data;

printf("B\_2060425\_BILHA P ABY\n");

printf(" --------------------------------\n");

printf("| IMPLEMENTATION OF LINKED LIST |\n");

printf(" --------------------------------\n");

do{

printf("\n1.INSERTION AT THE BEGINNING.\n");

printf("2.INSERTION AT THE END.\n");

printf("3.DISPLAY\n");

printf("4.DELETION FROM THE BEGINNING.\n");

printf("5.DELETION FROM THE END.\n");

printf("6. Exit\n");

printf("Select the option:");

scanf("%d", &i);

if (i == 1)

{

printf("Enter value of element:");

scanf("%d", &data);

insert\_begin(data);

}

else if (i == 2)

{

printf("Enter value of element:");

scanf("%d", &data);

insert\_end(data);

}

else if (i == 3)

display();

else if (i == 4)

delete\_begin();

else if (i == 5)

delete\_end();

else if (i == 6)

break;

else

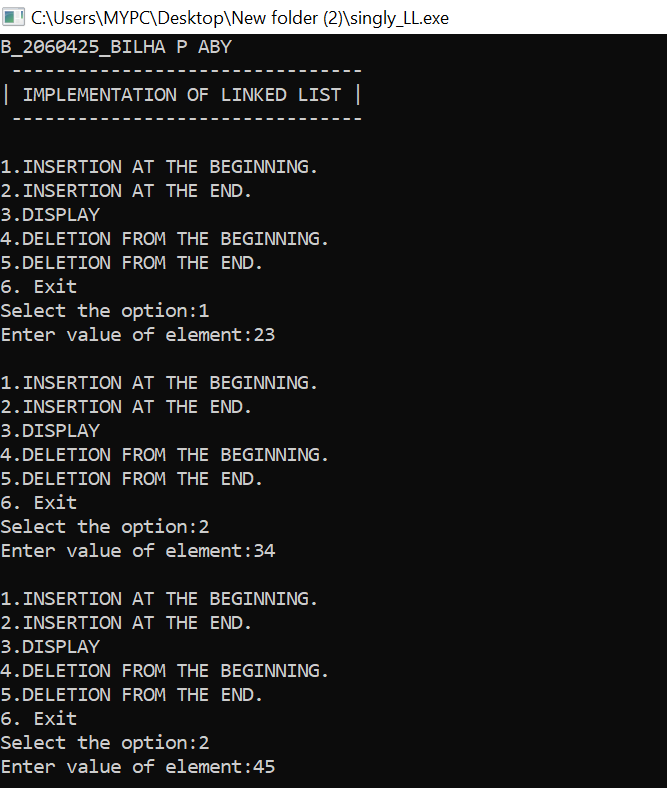
printf("INVALID OPTION!!.\n");

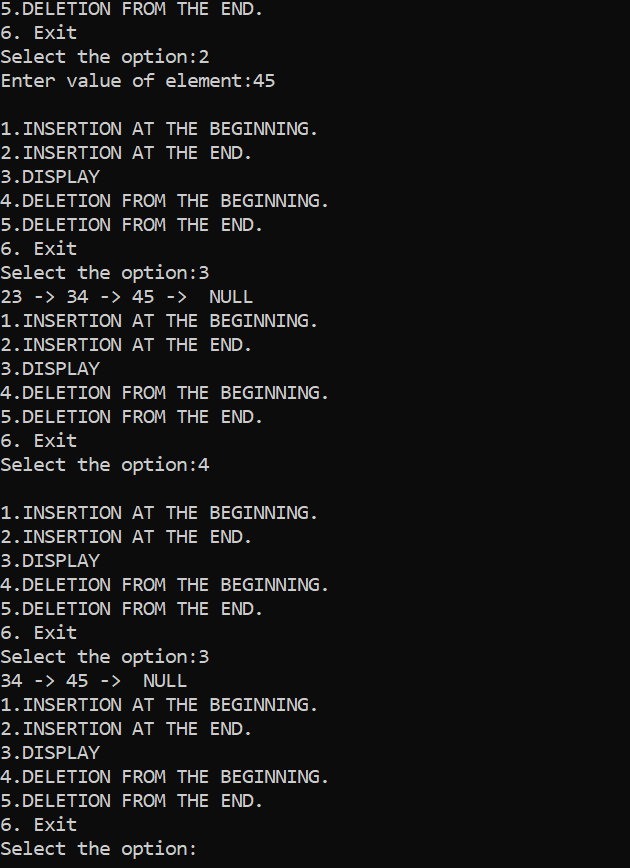
}while(i!=6);

return 0;

}

**OUTPUT:**





**RESULT:** singly linked list was implemented.

**EXPERIMENT NO 4**

**DATE: 06/10/2021**

**Linked List Implementation of Stack**

**AIM**: Implementing of Stack using Linked List.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

void push();

void pop();

void display();

struct node

{

int data;

struct node \*link;

};

struct node \*top=0;

int main()

{

int option;

printf("B\_2060425\_BILHA P ABY\n");

printf(" ---------------------------------------------\n");

printf("|STACK IMPLEMENTATION USING LINKED LIST |\n");

printf(" ---------------------------------------------\n");

do

{

printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n");

printf("\nEnter the option no. : ");

scanf("%d",&option);

switch(option)

{

case 1:

{

push ();

break;

}

case 2:

{

pop ();

break;

}case 3:

{

display();

break;

}

case 4:

{

break;

}

default:

{

printf ("\nInvalid option\n");

}}}

while(option!=4);

return 0;

}

void push ()

{

int x;

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

printf("Enter value to be pushed:");

scanf("%d",&x);

newnode->data=x;

newnode->link=top;

top=newnode;

}

void pop()

{

struct node \*temp;

temp=top;

if(top==0)

{

printf("-----EMPTY STACK-----");

}

else

{

printf("The popped element is %d", top->data);

top=top->link;

free(temp);

}

}

void display()

{

struct node \*temp;

temp=top;

if(top==0)

{

printf("-------EMPTY STACK-----");

}

else

{

while(temp!=0)

{

printf("%d\n", temp->data);

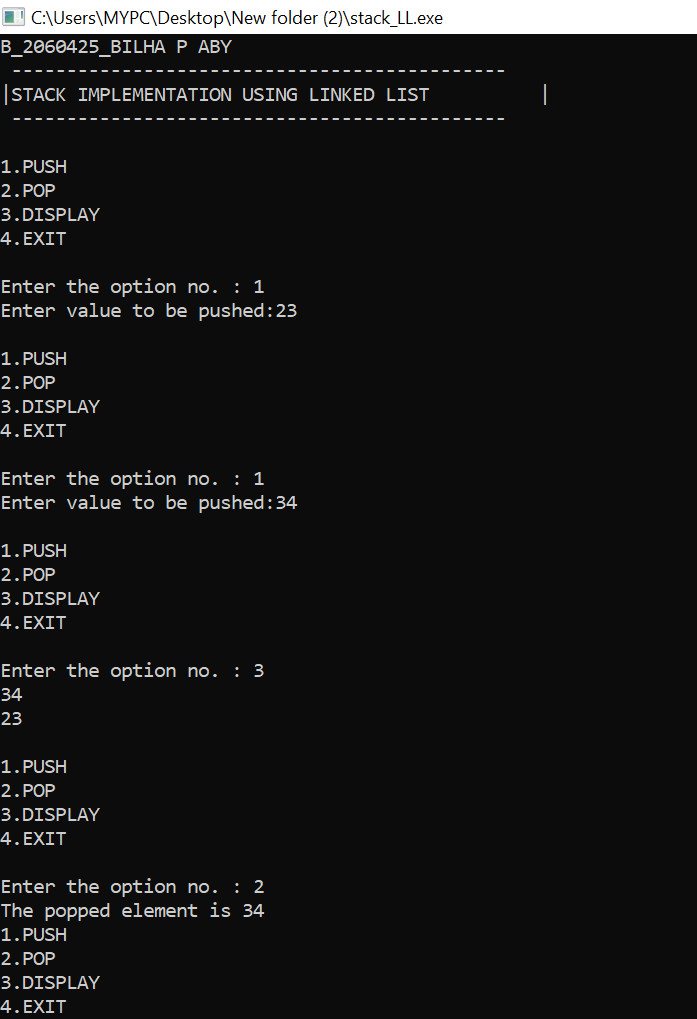
temp=temp->link;

}

}

}

**OUTPUT:**



**RESULT:** Stack was implemented using linked list.

**EXPERIMENT NO 5**

**DATE: 06/10/2021**

**Linked List Implementation of Queue**

**AIM**: Implementing of Queue using Linked List.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

void enqueue();

void display();

void dequeue();

struct node

{

int data;

struct node \*next;

};

struct node \*front=0;

struct node \*rear=0;

void enqueue()

{

int x;

struct node \*newnode;

newnode=(struct node\*)malloc(sizeof(struct node));

printf("enter value to be Enqueued:");

scanf("%d",&x);

newnode->data=x;

if(front==0)

{

front=newnode;

rear=newnode;

front->next=0;

rear->next=0;

}

else

{

rear->next=newnode;

rear=newnode;

rear-> next =0;

}

}

void display()

{

struct node \*temp;

if(front==0)

{

printf("---------EMPTY QUEUE-------- ");

}

else

{

temp=front;

while(temp!=0)

{

printf("\t%d |", temp->data);

temp=temp->next;

}

}

}

void dequeue()

{

if(front==0)

{

printf("---------EMPTY QUEUE-------- ");

}

else

{

struct node \*temp;

temp=front;

front=front->next;

printf("The Dequeued element is %d ",temp->data);

free(temp);

}

}

int main()

{

int option;

printf("B\_2060425\_BILHA P ABY\n");

printf(" ---------------------------------------------\n");

printf("|QUEUE IMPLEMENTATION USING LINKED LIST |\n");

printf(" ---------------------------------------------\n");

do

{

printf("\n1.ENQUEUE\n2.DEQUEUE\n3.DISPLAY\n4.EXIT\n");

printf("\nEnter the option no. : ");

scanf("%d",&option);

switch(option)

{

case 1:

{

enqueue();

break;

}

case 2:

{

dequeue();

break;

}

case 3:

{

display();

break;

}

case 4:

{

break;

}

default:

{

printf ("\nInvalid option\n");

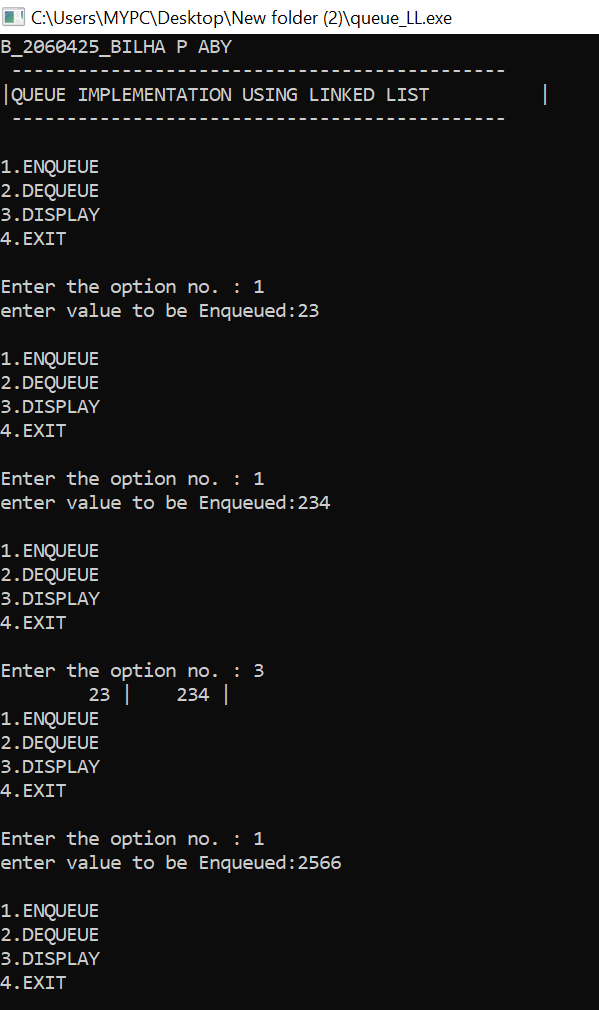
}}}

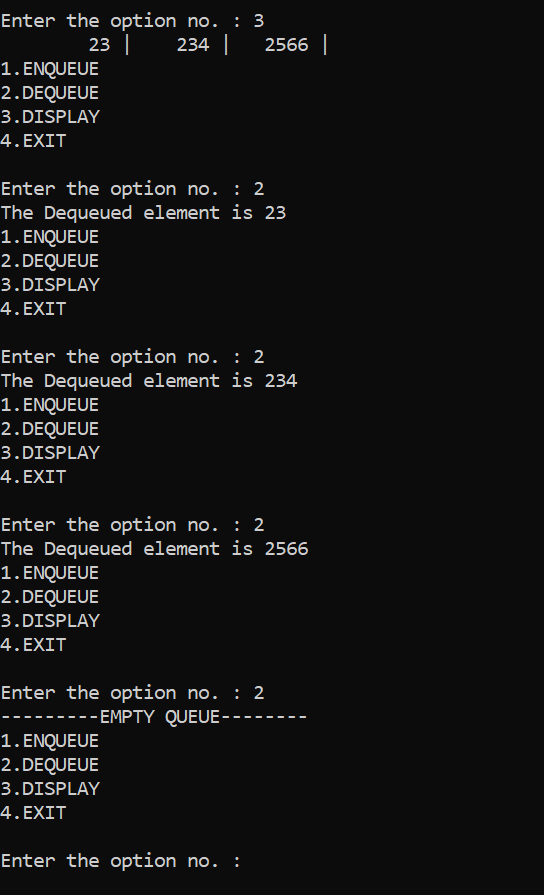
while(option!=4);

return 0;

}

**OUTPUT:**





**RESULT:** Queue was implemented using linked list

**EXPERIMENT NO 6**

**DATE: 20/10/2021**

**Application of Stack- Infix to Postfix**

**AIM**: Infix to postfix

**PROGRAM:**

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("NAME: BILHA P ABY\n");

printf("REG NO. : 2060425\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

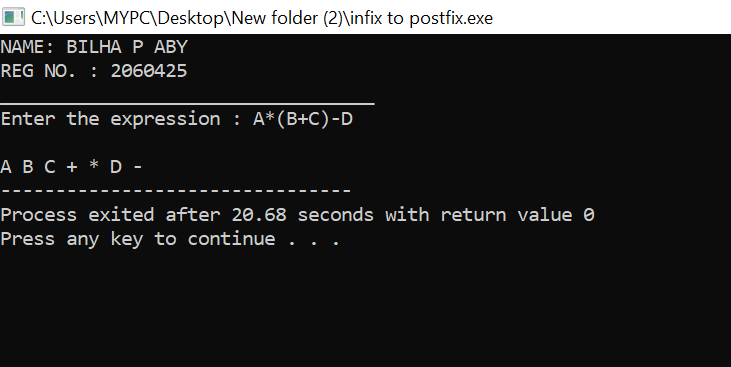
{

printf("%c ",pop());

}return 0;

}

**OUTPUT:**



**RESULT:** Infix to postfix is evaluated.

**EXPERIMENT NO 6b**

**DATE: 20/10/2021**

**Application of Stack- Postfix Evaluation**

**AIM**: Postfix evaluation.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

int top = 10;

struct node

{

char ch;

struct node \*next;

struct node \*prev;

} \*stack[11];

typedef struct node node;

void push(node \*str)

{

if (top <= 0)

printf("Stack is Full ");

else

{

stack[top] = str;

top--;

}

}

node \*pop()

{

node \*exp;

if (top >= 10)

printf("Stack is Empty ");

else

exp = stack[++top];

return exp;

}

void convert(char exp[])

{

node \*op1, \*op2;

node \*temp;

int i;

for (i=0;exp[i]!='\0';i++)

if (exp[i] >= 'a'&& exp[i] <= 'z'|| exp[i] >= 'A' && exp[i] <= 'Z')

{

temp = (node\*)malloc(sizeof(node));

temp->ch = exp[i];

temp->next = NULL;

temp->prev = NULL;

push(temp);

}

else if (exp[i] == '+' || exp[i] == '-' || exp[i] == '\*' || exp[i] == '/' ||

exp[i] == '^')

{

op1 = pop();

op2 = pop();

temp = (node\*)malloc(sizeof(node));

temp->ch = exp[i];

temp->next = op1;

temp->prev = op2;

push(temp);

}

}

void display(node \*temp)

{

if (temp != NULL)

{

display(temp->prev);

printf("%c", temp->ch);

display(temp->next);

}

}

int main()

{

char exp[50];

printf("NAME: BILHA P ABY\n");

printf("REG NO. : 2060425\n");

printf("Enter the postfix expression : ");

scanf("%s", exp);

convert(exp);

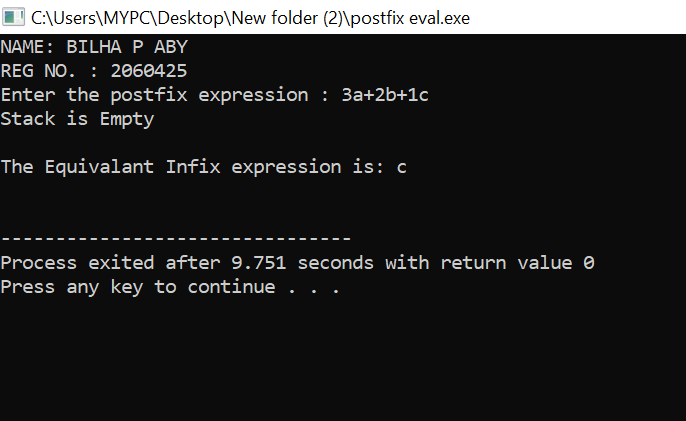
printf("\n\nThe Equivalant Infix expression is: ");

display(pop());

printf("\n\n");

}

**OUTPUT:**



**RESULT:** Postfix expression was evaluated.

**EXPERIMENT NO 7**

**DATE: 24/11/2021**

**BINARY SEARCH TREE**

**AIM**: Implementation of BST

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*left,\*right;

};

struct node \*create()

{

int x;

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

printf("\nEnter The data(-1 for no node) : ");

scanf("%d",&x);

if(x==-1)

{

return 0;

}

newnode->data=x;

printf("Enter left child of %d",x);

newnode->left=create();

printf("Enter Right child of %d",x);

newnode->right=create();

return newnode;

}

void Inorder(struct node\* root)

{

if(root==NULL)

return;

Inorder(root->left);

printf("%d-> ",root->data);

Inorder(root->right);

}

void Preorder(struct node\* root)

{

if(root==NULL)

return;

printf("%d-> ",root->data);

Preorder(root->left);

Preorder(root->right);

}

void Postorder(struct node\* root)

{

if(root==NULL)

return;

Postorder(root->left);

Postorder(root->right);

printf("%d-> ",root->data);

}

struct node \*search(struct node \*root, int data)

{

if(root == NULL)

printf("\nElement not found");

else if(data < root->data)

{

root->left=search(root->left, data);

}

else if(data > root->data)

{

root->right=search(root->right, data);

}

else

printf("\nElement found is: %d", root->data);

return root;

}

int main()

{

int ch,data;

struct node \*root;

root=0;

do

{

printf("\nEnter your choice :\n1.Create\n2.Inorder\n3.Preorder\n4.Postorder\n5.Searching\n6.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

root=create();

break;

case 2:

printf("Inorder Traversal\n");

Inorder(root);

break;

case 3: printf("Preorder Traversal\n");

Preorder(root);

break;

case 4: printf("Postorder Traversal\n");

Postorder(root);

break;

case 5:

printf("\nEnter the element to search: ");

scanf("%d", &data);

root=search(root, data);

break;

case 6:

break;

default:

printf("\nInvalid choice\n" );

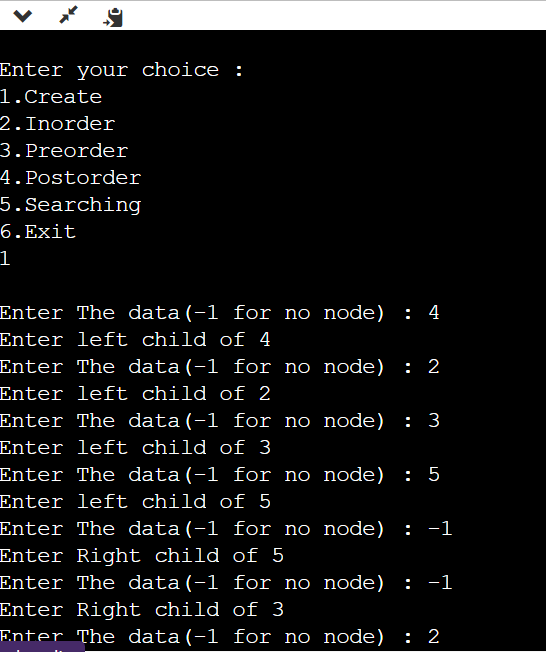
}

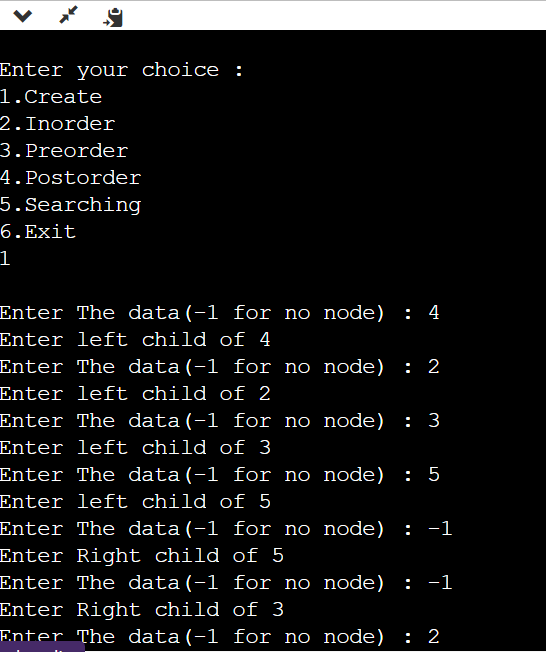
}while(ch!=6);

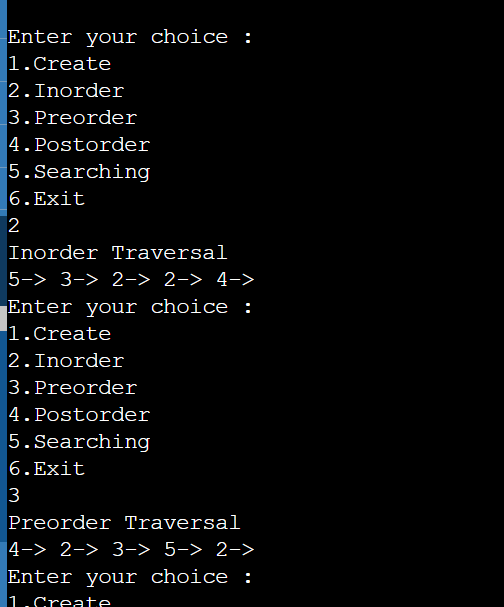
return 0;

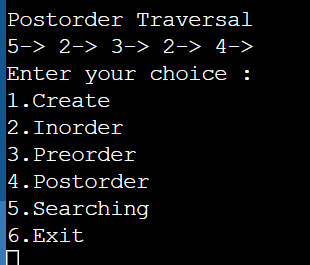
}

**OUTPUT:**









**RESULT**: Binary search tree was implemented successfully.

**EXPERIMENT NO 8**

**DATE: 1/12/2021**

**Hashing Technique**

**AIM**: To implement linear and separate chain hashing.

**PROGRAM:**

//Linear Hashing

#include<stdio.h>

#include<stdlib.h>

#define table\_size 6

int h[table\_size]= {NULL};

void insert()

{int key,index,i,flag=0,hkey;

printf("Enter a value to insert into hash table: ");

scanf("%d",&key);

hkey=key%table\_size;

for(i=0;i<table\_size;i++)

{

index=(hkey+i)%table\_size;

if(h[index] == NULL)

{

h[index]=key;

break;

} }

if(i == table\_size)

printf("Element cannot be inserted \n");

}void search()

{

int key,index,i,flag=0,hkey;

printf("Enter search element: ");

scanf("%d",&key);

hkey=key%table\_size;

for(i=0;i<table\_size; i++)

{

index=(hkey+i)%table\_size;

if(h[index]==key)

{

printf("\nvalue is found at index %d \n",index);

break;

}}

if(i == table\_size)

printf("\nvalue is not found\n");

}void display()

{

int i;

printf("Elements in the hash table are: ");

for(i=0;i<table\_size;i++)

printf("\n at index %d \t value = %d \n",i,h[i]);

}void main()

{

int opt,i;

printf("1.Insert\n");

printf("2.Display\n");

printf("3.Search\n");

printf("4.Exit\n");

while(1)

{ printf("Enter your choice: ");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

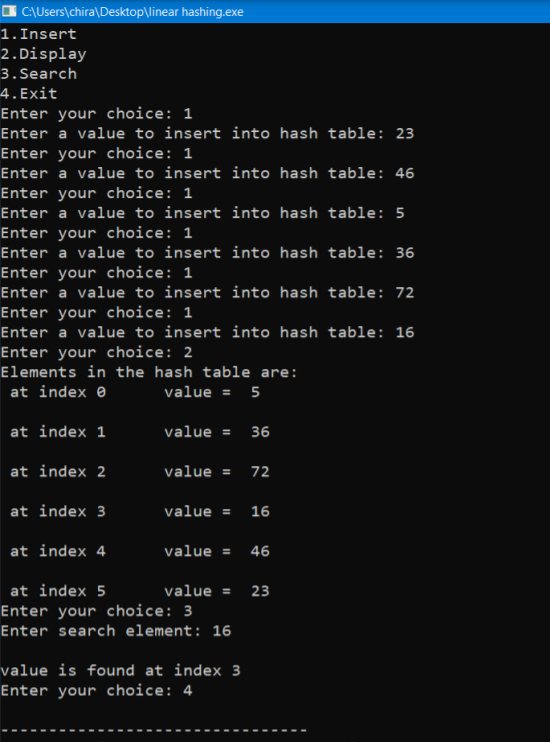
exit(0);

default:

printf("Enter correct choice \n");

}}}

**OUTPUT:**

****

**RESULT:** Linear hashing was implemented.

**Program:**

#include <stdio.h>

#include <stdlib.h>

#define table\_size 6

struct node

{

int data;

struct node \*next;

};

struct node \*head[table\_size]={NULL},\*c;

void insert()

{

int i,key;

printf("Enter a value to insert into hash table: ");

scanf("%d",&key);

i=key%table\_size;

struct node \* newnode=(struct node \*)malloc(sizeof(struct node));

newnode->data=key;

newnode->next = NULL;

if(head[i] == NULL)

head[i] = newnode;

else

{

c=head[i];

while(c->next != NULL)

{

c=c->next;

}

c->next=newnode;

}

}

void search()

{

int key,index;

printf("Enter the element to be searched:");

scanf("%d",&key);

index=key%table\_size;

if(head[index] == NULL)

printf("Search element not found\n");

else

{

for(c=head[index];c!=NULL;c=c->next)

{

if(c->data == key)

{

printf("Search element found at index: %d ",index);

break;

}

}

if(c==NULL)

printf("Search element not found\n");

}

}

void display()

{

int i;

for(i=0;i<table\_size;i++)

{

printf("\nEntries at index %d\n",i);

if(head[i] == NULL)

{

printf("No Hash Entry");

}

else

{

for(c=head[i];c!=NULL;c=c->next)

printf("%d-> ",c->data);

}

}

}

void main()

{

int opt,key,i;

printf("1.Insert\n");

printf("2.Display\n");

printf("3.Search\n");

printf("4.Exit\n");

while(1)

{ printf("\nEnter your choice: ");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:

exit(0);

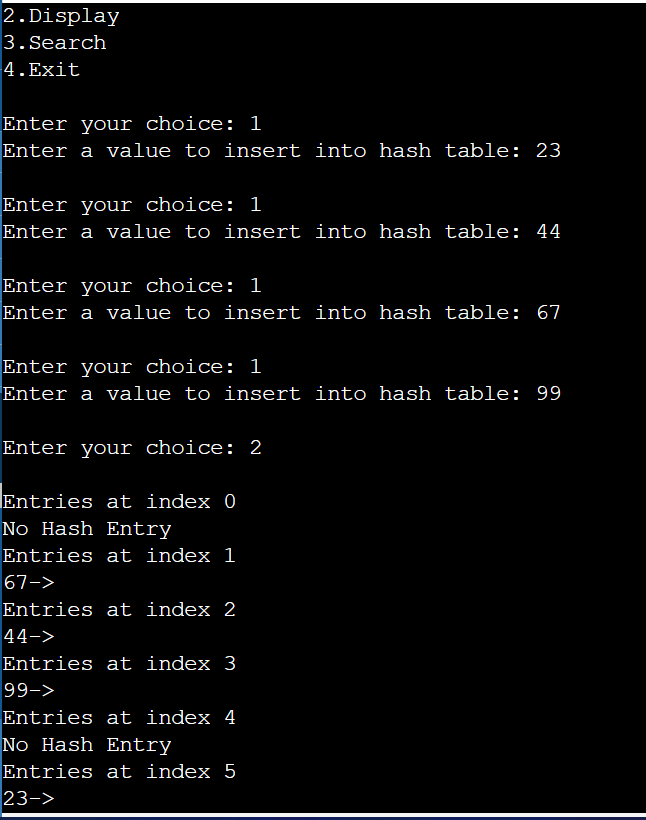
default:

printf("Enter correct choice \n");

}

}

}



**Result**: separate chain hashing was i

**EXPERIMENT NO 9**

**DATE: 8/12/2021**

**SORTING**

**AIM**: To implement merge and quick sort.

**PROGRAM:**

#include<stdio.h>

void mergesort(int a[],int i,int j);

void merge(int a[],int i1,int j1,int i2,int j2);

int main()

{

int a[30],n,i;

printf("Bilha P Aby\_2060425\n");

printf("Enter no of elements:");

scanf("%d",&n);

printf("Enter array elements:");

for(i=0;i<n;i++)

scanf("%d",&a[i]);

mergesort(a,0,n-1);

printf("\nSorted array is :");

for(i=0;i<n;i++)

printf("%d ",a[i]);

return 0;

}

void mergesort(int a[],int i,int j)

{

int mid;

if(i<j)

{

mid=(i+j)/2;

mergesort(a,i,mid); //left recursion

mergesort(a,mid+1,j); //right recursion

merge(a,i,mid,mid+1,j); //merging of two sorted sub-arrays

}

}

void merge(int a[],int i1,int j1,int i2,int j2)

{

int temp[50]; //array used for merging

int i,j,k;

i=i1; //beginning of the first list

j=i2; //beginning of the second list

k=0;

while(i<=j1 && j<=j2) //while elements in both lists

{

if(a[i]<a[j])

temp[k++]=a[i++];

else

temp[k++]=a[j++];

}

while(i<=j1) //copy remaining elements of the first list

temp[k++]=a[i++];

while(j<=j2) //copy remaining elements of the second list

temp[k++]=a[j++];

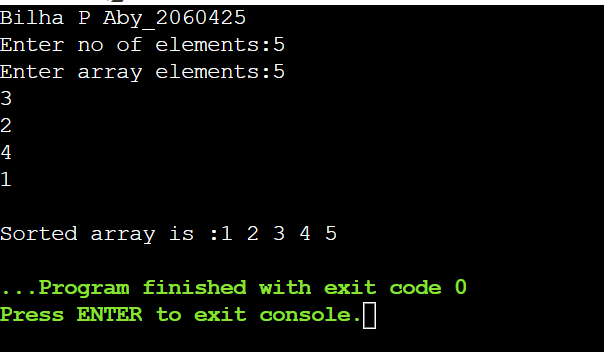
//Transfer elements from temp[] back to a[]

for(i=i1,j=0;i<=j2;i++,j++)

a[i]=temp[j];

}

**Output**



**PROGRAM:**

#include<stdio.h>

void quicksort(int number[25],int first,int last){

int i, j, pivot, temp;

if(first<last){

pivot=first;

i=first;

j=last;

while(i<j){

while(number[i]<=number[pivot]&&i<last)

i++;

while(number[j]>number[pivot])

j--;

if(i<j){

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

temp=number[pivot];

number[pivot]=number[j];

number[j]=temp;

quicksort(number,first,j-1);

quicksort(number,j+1,last);

}

}

int main(){

int i, count, number[25];

printf("Bilha P Aby\_2060425\n");

printf("How many elements are u going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

for(i=0;i<count;i++)

scanf("%d",&number[i]);

quicksort(number,0,count-1);

printf("Order of Sorted elements: ");

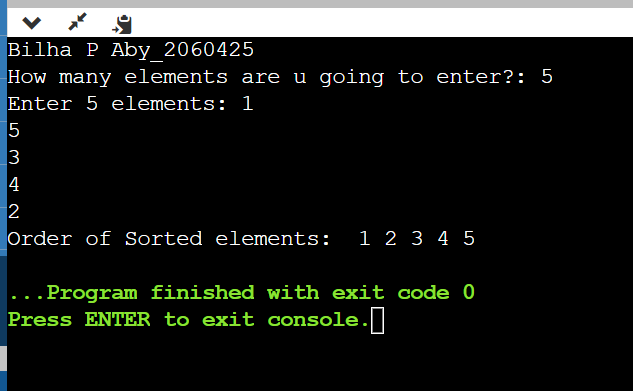
for(i=0;i<count;i++)

printf(" %d",number[i]);

return 0;

}

**OUTPUT:**



**RESULT:** Merge sort and quick sort was implemented successfully.

**EXPERIMENT NO 10**

**DATE: 15/12/2021**

**SHORTEST PATH ALGORITHM**

**AIM**: To Implement Dijkstra Algorithm.

**PROGRAM:**

#include<stdio.h>

void dijkstra(int G[10][10],int n,int start);

int main()

{

int G[10][10],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);

return(0);

}

void dijkstra(int G[10][10],int n,int start)

{

int cost[10][10],distance[10],pred[10];

int visited[10],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]=999;

else

cost[i][j]=G[i][j];

//initialize pred[],distance[] and visited[]

for(i=0;i<n;i++)

{

distance[i]=cost[start][i];

pred[i]=start;

visited[i]=0;

}

distance[start]=0;

visited[start]=1;

count=1;

while(count<n-1)

{

mindistance=999;

//nextnode at minimum distance

for(i=0;i<n;i++)

if(distance[i]<mindistance&&!visited[i])

{

mindistance=distance[i];

nextnode=i;

}

//better path or not

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!=start)

{

printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{

j=pred[j];

printf("<-%d",j);

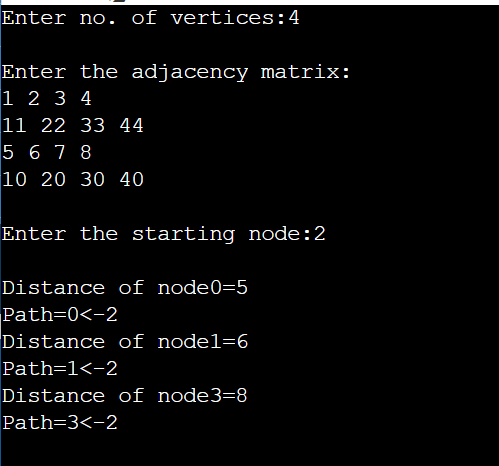
}

while(j!=start);

}

}

**Output:**

****

**RESULT:** Shortest path algorithm was implemented successfully.