**STACK USING ARRAY**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

#define MAX 100

int st[MAX];

int top = -1;

void push(int);

void pop();

void peek();

void display();

int main(){

while(1){

printf("\n1.push 2.pop 3.peek 4.display 5.exit\n");

print("enter your choice: ");

int ch, num;

scanf("%d", &ch);

switch(ch){

case 1:

printf("\nenter the number : ");

scanf("%d", &num);

push(num);

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5:

exit(0);

}

}

}

void push(int n){

if( top == MAX-1 ){

printf("\nstack over flow");

}else{

st[++top] = n;

printf("\n%d has been pushed", n);

}

}

void pop(){

if( top == -1 ){

printf("\nstack under flow");

}else{

printf("\n%d has been popped", st[top--]);

}

}

void peek(){

if( top == -1 ){

printf("\nstack under flow");

}else{

printf("\npeek element is : %d", st[top]);

}

}

void display(){

if( top == -1 ){

printf("\nstack under flow");

}else{

int i;

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

printf("\n[ %d ]", st[i]);

}

}

}

**OUTPUT**

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 1

enter the number : 10

10 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 1

enter the number : 20

20 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 1

enter the number : 30

30 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 1

enter the number : 40

40 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 2

40 has been popped

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 3

peek element is : 30

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 4

[ 30 ]

[ 20 ]

[ 10 ]

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 1

enter the number : 70

70 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

enter your choice: 4

[ 70 ]

[ 30 ]

[ 20 ]

[ 10 ]

**RESULT:**

Stack using array was executed successfully

**STACK USING LINKED LIST**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

void push(int n);

void pop();

void peek();

void display();

struct Node{

struct Node \*next;

int val;

}\*top;

int main(){

while( 1 ){

printf("\n1.push 2.pop 3.peek 4.display 5.exit\n");

printf("Enter your choice: ");

int num, ch;

scanf("%d", &ch);

switch(ch){

case 1:

printf("\nenter number : ");

scanf("%d", &num);

push(num);

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5:

exit(0);

}

}

}

void push(int n){

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

newnode->val = n;

if( top == NULL ){

newnode->next = NULL;

}else{

newnode->next = top;

}

top = newnode;

printf("\n%d has been pushed", n);

}

void pop(){

struct Node \*temp = top;

if(temp==NULL){

printf("stack under flow");

return;

}

printf("\n%d has been popped", temp->val);

top = top->next;

free(temp);

}

void peek(){

if(top==NULL){

printf("stack under flow");

return;

}

printf("peek element is : %d", top->val);

}

void display(){

struct Node \*cur = top;

if(cur==NULL){

printf("stack under flow");

return;

}

while( cur != NULL ){

printf("[ %d ]\n", cur->val);

cur = cur->next;

}

}

**OUTPUT**

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 1

enter number : 10

10 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 1

enter number : 20

20 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 1

enter number : 30

30 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 4

[ 30 ]

[ 20 ]

[ 10 ]

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 2

30 has been popped

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 60

60 has been pushed

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 3

peek element is : 60

1.push 2.pop 3.peek 4.display 5.exit

Enter your choice: 4

[ 60 ]

[ 20 ]

[ 10 ]

**RESULT:**

Stack using linked list was executed successfully

**QUEUE USING ARRAY**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

int rear = 0, front = 0;

int ar[MAX];

void enqueue(int val){

if(rear==MAX){

printf("queue over flow");

return;

}

ar[rear++] = val;

}

int dequeue(){

if(front==rear){

printf("queue under flow");

return -1;

}

return ar[front++];

}

void show(){

if(front==rear){

printf("queue under flow");

return ;

}

for(int i=front; i<rear; i++) printf("%d ", ar[i]);

}

int main(){

int num, ch;

while(!0){

printf("\n1.enqueue 2.dequeue 3.show 4.exit\n");

printf("enter your choice: ");

scanf("%d", &ch);

switch(ch){

case 1:

printf("\nenter the number: ");

scanf("%d", &num);

enqueue(num);

break;

case 2:

printf("\n%d has been deleted",dequeue());

break;

case 3:

show();

break;

case 4:

exit(0);

}

}

}

**OUTPUT**

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 1

enter the number: 100

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 1

enter the number: 200

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 1

enter the number: 300

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 3

100 200 300

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 2

100 has been deleted

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 3

200 300

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 1

enter the number: 400

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 1

enter the number: 500

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 3

200 300 400 500

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 2

200 has been deleted

1.enqueue 2.dequeue 3.show 4.exit

enter your choice: 3

300 400 500

**RESULT:**

Queue using array was executed successfully

**SINGLY LINKED LIST**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

int size;

struct Node{

struct Node\* next;

int val;

} \*head = NULL;

void insertAtFirst(int val){

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

newnode->next = NULL;

newnode->val = val;

if(head==NULL){

head = newnode;

return;

}

newnode->next = head;

head = newnode;

size++;

return;

}

void insertAtEnd(int val){

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

newnode->next = NULL;

newnode->val = val;

if(head==NULL){

head = newnode;

return;

}

struct Node\* cur = head;

while(cur->next!=NULL) cur = cur->next;

cur->next = newnode;

size++;

}

void insertAtMid(int val){

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

newnode->next = NULL;

newnode->val = val;

if(head==NULL){

head = newnode;

return;

}

int mid = size/2;

struct Node\* cur = head;

for(int i=0; i<mid; i++) cur = cur->next;

newnode->next = cur->next;

cur->next = newnode;

size++;

}

int deleteAtFirst(){

struct Node\* temp = head;

int res = temp->val;

head = temp->next;

temp = NULL;

free(temp);

return res;

}

int deleteAtLast(){

struct Node\* cur = head;

while(cur->next->next!=NULL) cur = cur->next;

int res = cur->next->val;

cur->next = NULL;

free(cur->next);

return res;

}

void show(){

struct Node\* cur = head;

while(cur!=NULL){

printf("%d -> ", cur->val);

cur = cur->next;

}

}

int main(){

size = 0;

int ch, num;

while(1){

printf("\n1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit");

scanf("%d", &ch);

switch(ch){

case 1:

printf("\nenter the element: ");

scanf("%d", &num);

insertAtFirst(num);

break;

case 2:

scanf("%d", &num);

insertAtEnd(num);

break;

case 3:

scanf("%d", &num);

insertAtMid(num);

break;

case 4:

printf("%d deleted succesfully\n", deleteAtFirst());

break;

case 5:

printf("%d deleted succesfully\n", deleteAtLast());

break;

case 6:

show();

break;

case 7:

exit(1);

}

}

return 0;

}

**OUTPUT**

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit1

enter the element: 111

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit1

enter the element: 112

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit1

enter the element: 113

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit1

enter the element: 114

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit6

114 -> 113 -> 112 -> 111 ->

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit2

489

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit6

114 -> 113 -> 112 -> 111 -> 489 ->

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit3

546

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit6

114 -> 113 -> 112 -> 546 -> 111 -> 489 ->

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit5

489 deleted succesfully

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit6

114 -> 113 -> 112 -> 546 -> 111 ->

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit4

114 deleted succesfully

1.Insert at first 2.Insert at end 3.Insert at middle 4.Delete at first 5.Delete at end 6.Show 7.Exit6

113 -> 112 -> 546 -> 111 ->

**RESULT:**

Singly linked list was executed successfully

**POLYNOMIAL ADDITION**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

struct Node{

int coeff;

int pow;

struct Node \*next;

};

void create(int coeff, int pow, struct Node\*\* head){

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

newnode->coeff = coeff;

newnode->pow = pow;

newnode->next = NULL;

if(\*head==NULL){

\*head = newnode;

return;

}

struct Node\* cur = \*head;

while(cur->next!=NULL){

cur = cur->next;

}

cur->next = (struct Node\*)malloc(sizeof(struct Node));

cur->next = newnode;

}

struct Node\* add(struct Node\* temp1, struct Node\* temp2){

struct Node\* out = NULL;

while(temp1&&temp2){

if(temp1->pow>temp2->pow){

create(temp1->coeff, temp1->pow, &out);

temp1 = temp1->next;

}else if(temp1->pow<temp2->pow){

create(temp2->coeff, temp2->pow, &out);

temp2 = temp2->next;

}else{

create(temp1->coeff+temp2->coeff, temp1->pow, &out);

temp1 = temp1->next;

temp2 = temp2->next;

}

}

while(temp1||temp2){

if(temp1){

create(temp1->coeff, temp1->pow, &out);

temp1 = temp1->next;

}

if(temp2){

create(temp2->coeff, temp2->pow, &out);

temp2 = temp2->next;

}

}

return out;

}

void print(struct Node\* head){

struct Node\* cur = head;

while(cur!=NULL){

printf(" %dx^%d +", cur->coeff, cur->pow);

cur = cur->next;

}

}

int main(){

struct Node \*poly1 = NULL;

struct Node \*poly2 = NULL;

struct Node \*res = NULL;

create(2, 5, &poly1);

create(5, 4, &poly1);

create(6, 3, &poly1);

create(8, 2, &poly1);

create(12, 1, &poly1);

create(34, 0, &poly1);

create(2, 4, &poly2);

create(7, 3, &poly2);

create(7, 2, &poly2);

create(3, 1, &poly2);

printf("\n");

print(poly1);

printf("\n");

print(poly2);

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

res = add(poly1, poly2);

print(res);

return 0;

}

OUTPUT

2x^5 + 5x^4 + 6x^3 + 8x^2 + 12x^1 + 34x^0 +

2x^4 + 7x^3 + 7x^2 + 3x^1 +

----------------------------------------

2x^5 + 7x^4 + 13x^3 + 15x^2 + 15x^1 + 34x^0 +

**RESULT:**

Polynomial Addition was executed successfully

**BINARY SEARCH**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

int search(int\*, int, int, int);

int main(){

int n;

scanf("%d", &n);

int\* ar = (int\*)malloc(n\*sizeof(int));

for(int i=0; i<n; i++) scanf("%d", ar+i);

int target;

scanf("%d", &target);

int index = search(ar, target, 0, n);

if(index==-1) printf("not found");

else printf("element found in %d index", (index+1));

}

int search(int\* ar, int t, int s, int e){

if(s<e){

int mid = (s+e)/2;

if(\*(ar+mid)==t) return mid;

else if(\*(ar+mid)<t) return search(ar, t, mid+1, e);

else return search(ar, t, s, mid-1);

}

return -1;

}

OUTPUT

10

20 34 37 45 56 58 67 77 89 99

58

element found in 6 index

**BINARY SEARCH TREE AND TRAVERSAL**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

struct Node{

int val;

struct Node\* right;

struct Node\* left;

};

struct Node\* create(int val){

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

newnode->val = val;

newnode->right = NULL;

newnode->left = NULL;

return newnode;

}

struct Node\* insert(struct Node\* root, int val){

if(root == NULL){

return create(val);

}else{

if(root->val>=val){

root->left = insert(root->left, val);

}else{

root->right = insert(root->right, val);

}

}

return root;

}

void preorder(struct Node\* root){

if(root != NULL){

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

preorder(root->left);

preorder(root->right);

}

}

void inorder(struct Node \*root){

if(root != NULL){

inorder(root->left);

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

inorder(root->right);

}

}

void postorder(struct Node \*root){

if(root != NULL){

postorder(root->left);

postorder(root->right);

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

}

}

int main()

{

struct Node\* root = NULL;

int n;

scanf("%d", &n);

printf("\nenter the elements: ");

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

int num;

scanf("%d", &num);

root = insert(root, num);

}

printf("PREORDER\n");

preorder(root);

printf("\n");

printf("INORDER\n");

inorder(root);

printf("\n");

printf("postORDER\n");

postorder(root);

}

**OUTPUT**

10

enter the elements: 45 67 23 12 10 78 78 8 5 98

PREORDER

45 23 12 10 8 5 67 78 78 98

INORDER

5 8 10 12 23 45 67 78 78 98

POSTORDER

5 8 10 12 23 78 98 78 67 45

**RESULT:**

Binary Search Tree was executed successfully

**QUICK SORT**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

int partition(int ar[], int s, int e){

int pivot = ar[e];

int i, pind = s;

for(i=s; i<e; i++){

if(ar[i]<=pivot){

int t = ar[i];

ar[i] = ar[pind];

ar[pind] = t;

pind++;

}

}

int t = ar[e];

ar[e] = ar[pind];

ar[pind] = t;

return pind;

}

void sort(int ar[], int s, int e){

if(s>=e) return;

int pind = partition(ar, s, e);

sort(ar, s, pind-1);

sort(ar, pind+1, e);

}

int main()

{

int n;

scanf("%d", &n);

int ar[n];

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

sort(ar, 0, n);

printf("\nSorted Array: ");

for(int i=0; i<n; i++) printf("%d ", ar[i]);

}

OUTPUT

10

Enter the element:

4

6

5

1

23

45

12

78

22

90

Sorted Array: 1 4 5 6 12 22 23 45 78 90

**RESULT:**

Quick sort was executed successfully

**INSERTION SORT**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

int main(){

int n;

scanf("%d", &n);

int ar[n];

printf("enter the elements: ");

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

for(int i=1; i<n; i++){

int j=i-1;

int key = ar[i];

while(j>=0&&ar[j]>key){

ar[j+1] = ar[j];

j--;

}

ar[j+1] = key;

}

for(int i=0; i<n; i++) printf("%d ", ar[i]);

return 0;

}

**OUTPUT**

Enter the size of array : 10

enter the elements: 4 5 1 2 8 7 9 10 3 6

1 2 3 4 5 6 7 8 9 10

**RESULT:**

Insertion sort was executed successfully

**SELECTION SORT**

**PROGRAM**

#include<stdio.h>

int main(){

int n;

printf("enter the size : ");

scanf("%d", &n);

printf("\nenter the elements : ");

int ar[n], i, j;

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

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

int min\_ind = i;

for(j=i+1; j<n; j++)

if(ar[min\_ind]>ar[j]) min\_ind = j;

int t = ar[i];

ar[i] = ar[min\_ind];

ar[min\_ind] = t;

}

printf("\nsorted :\n");

for(j=0; j<n; j++) printf("%d ", ar[j]);

return 0;

}

OUTPUT

enter the size : 10

enter the elements : 4 5 1 2 8 7 9 10 3 6

sorted :

1 2 3 4 5 6 7 8 9 10

**RESULT:**

Selection sort was executed successfully

**BUBBLE SORT**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

int main(){

int n;

printf("enter the size: ");

scanf("%d", &n);

int ar[n];

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

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

for(int j=0; j<n-i-1; j++)

if(ar[j]>ar[j+1]){

int t = ar[j];

ar[j] = ar[j+1];

ar[j+1] = t;

}

}

printf("Sorted: ");

for(int i=0; i<n; i++) printf("%d ", ar[i]);

return 0;

}

**OUTPUT**

enter the size: 10

enter the elements:

4 5 1 2 8 7 9 10 3 6

Sorted: 1 2 3 4 5 6 7 8 9 10

**RESULT:**

Bubble sort was executed successfully

**CIRCULAR QUEUE USING ARRAY**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

# define max 5

int queue[max];

int front=-1;

int rear=-1;

void enqueue(int element){

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

front=0;

rear=0;

queue[rear]=element;

}else if((rear+1)%max==front) printf("\nQueue is overflow");

else {

rear=(rear+1)%max;

queue[rear]=element;

}

}

int dequeue(){

if((front==-1) && (rear==-1)) printf("\nQueue is underflow");

else if(front==rear){

printf("\nThe dequeued element is %d", queue[front]);

front=-1;

rear=-1;

}else{

printf("\nThe dequeued element is %d", queue[front]);

front=(front+1)%max;

}

}

void display(){

int i=front;

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

printf("\n Queue is empty");

}else{

printf("\nElements in a Queue are :");

while(i<=rear) {

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

i=(i+1)%max;

}

}

}

int main() {

int ch=1,x;

while(!0) {

printf("\n1.insert 2.delete 3.show 4.exit\nenter your choice: ");

scanf("%d", &ch);

switch(ch){

case 1:

printf("enter the element: ");

scanf("%d", &x);

enqueue(x);

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

}

}

return 0;

}

**OUTPUT**

1.insert 2.delete 3.show 4.exit

enter your choice: 1

enter the element: 10

1.insert 2.delete 3.show 4.exit

enter your choice: 1

enter the element: 20

1.insert 2.delete 3.show 4.exit

enter your choice: 1

enter the element: 30

1.insert 2.delete 3.show 4.exit

enter your choice: 3

Elements in a Queue are :10,20,30,

1.insert 2.delete 3.show 4.exit

enter your choice: 2

The dequeued element is 10

1.insert 2.delete 3.show 4.exit

enter your choice: 3

Elements in a Queue are :20,30,

1.insert 2.delete 3.show 4.exit

enter your choice: 1

enter the element: 40

1.insert 2.delete 3.show 4.exit

enter your choice: 2

The dequeued element is 20

1.insert 2.delete 3.show 4.exit

enter your choice: 2

The dequeued element is 30

1.insert 2.delete 3.show 4.exit

enter your choice: 2

The dequeued element is 40

1.insert 2.delete 3.show 4.exit

enter your choice: 2

Queue is underflow

1.insert 2.delete 3.show 4.exit

enter your choice: 3

Queue is empty

**RESULT:**

Circular Queue using Array was executed successfully

**DEPTH FIRST TRAVERSAL**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

int visit[MAX]={0}, adj[MAX][MAX];

void addEdge(int s, int e){

adj[s][e] = 1;

}

void show(int n){

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

printf("\n%dth vertex is connected with: ", i);

for(int j=0; j<n; j++) if(adj[i][j]==1) printf("%d ", j);

}

}

void dfs(int s, int n){

printf("%d ", s);

visit[s] = 1;

for(int i=0; i<n; i++) if(adj[s][i]==1&&!visit[i]) dfs(i, n);

}

int main(){

int v, e;

printf("enter the number of vertices and edges: ");

scanf("%d %d", &v, &e);

printf("enter the source vertex and destination vertex: \n");

for(int i=0; i<e; i++){

int S, E;

scanf("%d %d", &S, &E);

addEdge(S, E);

}

show(v);

printf("\nDFS Traversal: ");

dfs(0, v);

return 0;

}

**OUTPUT**

enter the number of vertices and edges: 5 8

enter the source vertex and destination vertex:

0 1

0 3

1 2

1 3

2 4

2 3

3 4

4 0

0th vertex is connected with: 1 3

1th vertex is connected with: 2 3

2th vertex is connected with: 3 4

3th vertex is connected with: 4

4th vertex is connected with: 0

DFS Traversal: 0 1 2 3 4

**RESULT:**

Depth first traversal was executed successfully

**BREADTH FIRST TRAVERSAL**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

#define MAX 50

int adj[MAX][MAX];

int visit[MAX] = {0};

int q[MAX+MAX];

int front = 0, rear = 0;

void add(int n){

q[rear++] = n;

}

int poll(){

return q[front++];

}

int isEmpty(){

return front==rear;

}

void show(int n){

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

printf("\n%dth vertex is connect with: ", i);

for(int j=0; j<n; j++) if(adj[i][j]==1) printf("%d ", j);

}

}

void addEdge(int s, int e){

adj[s][e] = 1;

}

void bfs(int s, int n){

add(s);

visit[s] = 1;

while(!isEmpty()){

int t = poll();

printf("%d ", t);

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

if(adj[t][i]==1&&!visit[i]){

add(i);

visit[i] = 1;

}

}

}

int main(){

printf("enter the number of vertices and edges: ");

int v, e;

scanf("%d %d", &v, &e);

printf("enter the source vertex and destination vertex: \n");

for(int i=0; i<e; i++){

int S, E;

scanf("%d %d", &S, &E);

addEdge(S, E);

}

show(v);

printf("\nBFS Traversal: ");

bfs(0, v);

return 0;

}

**OUTPUT**

enter the number of vertices and edges: 5 8

enter the source vertex and destination vertex:

0 1

0 3

1 2

1 3

2 4

2 3

3 4

4 0

0th vertex is connect with: 1 3

1th vertex is connect with: 2 3

2th vertex is connect with: 3 4

3th vertex is connect with: 4

4th vertex is connect with: 0

BFS Traversal: 0 1 3 2 4

**RESULT:**

Breadth first traversal was executed successfully