**1.Write a program to implement linear queue using array.**

#include <stdio.h>

#include<stdlib.h>

#define MAX 50

void insert();

void delete();

void display();

int queue\_array[MAX];

int rear = - 1;

int front = - 1;

int main() {

int choice;

while (1) {

printf("1.Insert element to queue \n");

printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice) {

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

printf("Wrong choice \n");

} } }

void insert() {

int add\_item;if (rear == MAX - 1)

printf("Queue Overflow \n");

else {

if (front == - 1)

front = 0;

printf("Inset the element in queue : ");

scanf("%d", &add\_item);

rear = rear + 1;

queue\_array[rear] = add\_item;

}

}

void delete() {

if (front == - 1 || front > rear) {

printf("Queue Underflow \n");

return ;

}

else {

printf("Element deleted from queue is : %d\n", queue\_array[front]);

front = front + 1;

}

}

void display() {

int i;

if (front == - 1)

printf("Queue is empty \n");

else {

printf("Queue is : \n");

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

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

printf("\n");

}

}

**Output:**

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 1

Inset the element in queue : 10

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 1

Inset the element in queue : 20

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 1

Inset the element in queue : 30

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 3

Queue is :

10 20 30

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 4

1. **Write a program to implement linear queue using linked list.**

#include <stdio.h>

#include <stdlib.h>

struct node {

int info;

struct node \*ptr;

}\*front,\*rear,\*temp,\*front1;

int frontelement();

void enq(int data);

void deq();

void empty();void display();

void create();

void queuesize();

int count = 0;

void main() {

int no, ch, e;

printf("\n 1 - Enque");

printf("\n 2 - Deque");

printf("\n 3 - Front element");

printf("\n 4 - Empty");

printf("\n 5 - Exit");

printf("\n 6 - Display");

printf("\n 7 - Queue size");

create();

while (1) {

printf("\n Enter choice : ");

scanf("%d", &ch);

switch (ch) {

case 1:

printf("Enter data : ");

scanf("%d", &no);

enq(no);

break;

case 2:

deq();

break;

case 3:

e = frontelement();

if (e != 0)

printf("Front element : %d", e);

else

printf("\n No front element in Queue as queue is empty");

break;

case 4:

empty();

break;

case 5:

exit(0);case 6:

display();

break;

case 7:

queuesize();

break;

default:

printf("Wrong choice, Please enter correct choice ");

break;

}

}

}

void create() {

front = rear = NULL;

}

void queuesize() {

printf("\n Queue size : %d", count);

}

void enq(int data) {

if (rear == NULL) {

rear = (struct node \*)malloc(1\*sizeof(struct node));

rear->ptr = NULL;

rear->info = data;

front = rear;

}

else {

temp=(struct node \*)malloc(1\*sizeof(struct node));

rear->ptr = temp;

temp->info = data;

temp->ptr = NULL;

rear = temp;

}count++;

}

void display() {

front1 = front;

if ((front1 == NULL) && (rear == NULL)) {

printf("Queue is empty");

return;

}

while (front1 != rear) {

printf("%d ", front1->info);

front1 = front1->ptr;

}

if (front1 == rear)

printf("%d", front1->info);

}

void deq() {

front1 = front;

if (front1 == NULL) {

printf("\n Error: Trying to display elements from empty queue");

return;

}

else

if (front1->ptr != NULL) {

front1 = front1->ptr;

printf("\n Dequed value : %d", front->info);

free(front);

front = front1;

}

else {

printf("\n Dequed value : %d", front->info);

free(front);front = NULL;

rear = NULL;

}

count--;

}

int frontelement() {

if ((front != NULL) && (rear != NULL))

return(front->info);

else

return 0;

}

void empty() {

if ((front == NULL) && (rear == NULL))

printf("\n Queue empty");

else

printf("Queue not empty");

}

**Output:**

1 - Enque

2 - Deque

3 - Front element

4 - Empty

5 - Exit

6 - Display

7 - Queue size

Enter choice : 1

Enter data : 14

Enter choice : 1

Enter data : 85

Enter choice : 1

Enter data : 38

Enter choice : 3

Front element : 14

Enter choice : 6

14 85 38

Enter choice : 7

Queue size : 3

Enter choice : 2

Dequed value : 14

Enter choice : 6

85 38

Enter choice : 7

Queue size : 2

Enter choice : 4

Queue not empty

Enter choice : 5

1. **Write a program to implement circular queue.**

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#define size 5

int main() {

int arr[size],R=-1,F=0,te=0,ch,n,i,x;

for(;;) {

system("cls");

printf("1. Add\n");

printf("2. Delete\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter Choice: ");

scanf("%d",&ch);

switch(ch) {

case 1:

if(te==size) {

printf("Queue is full");

getch();

}else {

printf("Enter a number ");

scanf("%d",&n);

R=(R+1)%size;

arr[R]=n;

te=te+1;

}

break;

case 2:

if(te==0) {

printf("Queue is empty");

getch();

}

else {

printf("Number Deleted = %d",arr[F]);

F=(F+1)%size;

te=te-1;

getch();

}

break;

case 3:

if(te==0) {

printf("Queue is empty");

getch();

}

else {

x=F;

for(i=1; i<=te; i++) {

printf("%d ",arr[x]);

x=(x+1)%size;

}

getch();

}

break;

case 4:exit(0);

break;

default:

printf("Wrong Choice");

getch();

}

}

return 0;

}

**Output:**

1. Add

2. Delete

3. Display

4. Exit

Enter Choice: 3

10 15 18 20

1. **Write a program to implement multiple queues in a single array.**

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

int insq( int queue[MAX], int qno, int rear[], int limit[], int \*ele) {

if( rear[qno] == limit[qno] )

return(-1);

else {

rear[qno]++;

rear[qno] = rear[qno] + 1;

queue[ rear[qno] ] = \*ele;

return(1);

}

}

int delq( int queue[MAX], int qno, int front[], int rear[], int \*ele) {

if( front[qno] == rear[qno] )

return(-1);

else {

front[qno]++; //... front[qno] = front[qno] + 1;

\*ele = queue[ front[qno] ];

return(1);

}}

int main() {

int queue[MAX],ele;

int bott[10], limit[10], f[10], r[10];

int i, n,qno,size,ch,reply;

printf("How many queues you want to enter? : ");

scanf("%d", &n);

size = MAX / n;

bott[0] = -1;

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

bott[i] = bott[i-1] + size;

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

limit[i] = bott[i] + size;

}

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

f[i] = r[i] = bott[i];

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

printf("\tMenu");

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

printf("\n 1. Insert in element Queue");

printf("\n 2. Delete element from a Queue");

printf("\n 3. Exit \n");

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

while(1) {

printf("Choose operation : ");

scanf("%d", &ch);

switch(ch) {

case 1 :

printf("\nEnter logical queue number (0 to %d) : ", n-1);

scanf("%d", &qno);

printf("Element to be entered in queue number %d : ",qno);

scanf("%d", &ele);

reply = insq(queue, qno, r, limit, &ele);

if( reply == -1)

printf("Queue %d is full \n", qno);

else

printf("%d is inserted in queue number %d.\n\n", ele, qno);

break;case 2 :

printf("\nEnter logical queue number (0 to %d) : ", n-1);

scanf("%d", &qno);

reply = delq(queue, qno, f, r, &ele);

if( reply == -1)

printf("\n Queue %d is empty. \n", qno);

else

printf("%d is deleted from queue number %d \n\n",ele, qno);

break;

case 3 : exit(0);

default: printf("Invalid operation \n");

}

}

return 0;

}

**Output:**

How many queues you want to enter? : 2

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

Menu

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

1. Insert in element Queue

2. Delete element from a Queue

3. Exit

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

Choose operation : 1

Enter logical queue number (0 to 1) : 0

Element to be entered in queue number 0 : 20

20 is inserted in queue number 0.

Choose operation : 1

Enter logical queue number (0 to 1) : 0

Element to be entered in queue number 0 : 10

10 is inserted in queue number 0.

Choose operation : 1

Enter logical queue number (0 to 1) : 1

Element to be entered in queue number 1 : 5

5 is inserted in queue number 1.

Choose operation : 1

Enter logical queue number (0 to 1) : 1

Element to be entered in queue number 1 : 15

15 is inserted in queue number 1.

Choose operation : 2

Enter logical queue number (0 to 1) : 0

20 is deleted from queue number 0

Choose operation : 2

Enter logical queue number (0 to 1) : 1

5 is deleted from queue number 1

Choose operation : 3

1. **Write a program to implement ‘Dequeue’.**

# include<stdio.h>

# define MAX 5

int deque\_arr[MAX];

int left = -1;

int right = -1;

void insert\_right() {

int added\_item;

if((left == 0 && right == MAX-1) || (left == right+1)) {

printf("Queue Overflow\n");

return;}

if (left == -1){

left = 0;

right = 0;}

else

if(right == MAX-1)

right = 0;

else

right = right+1;

printf("Input the element for adding in queue : ");

scanf("%d", &added\_item);

deque\_arr[right] = added\_item ;

}

void insert\_left() {

int added\_item;

if((left == 0 && right == MAX-1) || (left == right+1)) {

printf("Queue Overflow \n");

return; }

if (left == -1){

left = 0;

right = 0;

}

else

if(left== 0)

left=MAX-1;

else

left=left-1;

printf("Input the element for adding in queue : ");

scanf("%d", &added\_item);

deque\_arr[left] = added\_item ;

}

void delete\_left() {

if (left == -1) {

printf("Queue Underflow\n");

return ;}

printf("Element deleted from queue is : %d\n",deque\_arr[left]);

if(left == right){

left = -1;

right=-1;

}

else

if(left == MAX-1)

left = 0;

else

left = left+1;

}

void delete\_right() {

if (left == -1) {

printf("Queue Underflow\n");

return ; }

printf("Element deleted from queue is : %d\n",deque\_arr[right]);

if(left == right) {

left = -1;

right=-1;

}

else

if(right == 0)

right=MAX-1;

else

right=right-1; }

void display\_queue() {

int front\_pos = left,rear\_pos = right;

if(left == -1) {

printf("Queue is empty\n");

return; }

printf("Queue elements :\n");

if( front\_pos <= rear\_pos ) {

while(front\_pos <= rear\_pos) {

printf("%d ",deque\_arr[front\_pos]);

front\_pos++; }

}

else {

while(front\_pos <= MAX-1) {

printf("%d ",deque\_arr[front\_pos]);

front\_pos++; }

front\_pos = 0;

while(front\_pos <= rear\_pos) {

printf("%d ",deque\_arr[front\_pos]);

front\_pos++;

}

}

printf("\n");

}

void input\_que() {

int choice;

do{

printf("1.Insert at right\n");

printf("2.Delete from left\n");

printf("3.Delete from right\n");

printf("4.Display\n");

printf("5.Quit\n");

printf("Enter your choice : ");

scanf("%d",&choice);switch(choice) {

case 1:

insert\_right();

break;

case 2:

delete\_left();

break;

case 3:

delete\_right();

break;

case 4:

display\_queue();

break;

case 5:

break;

default:

printf("Wrong choice\n");

}

}while(choice!=5);

}

void output\_que() {

int choice;

do{

printf("1.Insert at right\n");

printf("2.Insert at left\n");

printf("3.Delete from left\n");

printf("4.Display\n");

printf("5.Quit\n");

printf("Enter your choice : ");

scanf("%d",&choice);

switch(choice) {

case 1:

insert\_right();

break;

case 2:

insert\_left();

break;

case 3:

delete\_left();break;

case 4:

display\_queue();

break;

case 5:

break;

default:

printf("Wrong choice\n");

}

}while(choice!=5);

}

int main() {

int choice;

printf("1.Input restricted dequeue\n");

printf("2.Output restricted dequeue\n");

printf("Enter your choice : ");

scanf("%d",&choice);

switch(choice) {

case 1 :

input\_que();

break;

case 2:

output\_que();

break;

default:

printf("Wrong choice\n");

}

}

**Output:**

1.Input restricted dequeue

2.Output restricted dequeue

Enter your choice : 2

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 1

Input the element for adding in queue : 30

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 1

Input the element for adding in queue : 50

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 4

Queue elements :

30 50

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 5

**6.Write a program to implement priority queue.**

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

void insert\_by\_priority(int);

void delete\_by\_priority(int);

void create();

void check(int);

void display\_pqueue();

int pri\_que[MAX];

int front, rear;

void main() {

int n, ch;

printf("\n1 - Insert an element into queue");printf("\n2 - Delete an element from queue");

printf("\n3 - Display queue elements");

printf("\n4 - Exit");

create();

while (1) {

printf("\nEnter your choice : ");

scanf("%d", &ch);

switch (ch) {

case 1:

printf("\nEnter value to be inserted : ");

scanf("%d",&n);

insert\_by\_priority(n);

break;

case 2:

printf("\nEnter value to delete : ");

scanf("%d",&n);

delete\_by\_priority(n);

break;

case 3:

display\_pqueue();

break;

case 4:

exit(0);

default:

printf("\nChoice is incorrect, Enter a correct choice");

} } }

void create() {

front = rear = -1;

}

void insert\_by\_priority(int data) {

if (rear >= MAX - 1){

printf("\nQueue overflow no more elements can be inserted");

return;

}

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

front++;

rear++;

pri\_que[rear] = data;

return;

}

else

check(data);

rear++;

}

void check(int data) {

int i,j;

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

if (data >= pri\_que[i]) {

for (j = rear + 1; j > i; j--) {

pri\_que[j] = pri\_que[j - 1];

}

pri\_que[i] = data;

return;

}

}

pri\_que[i] = data;

}

void delete\_by\_priority(int data) {

int i;

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

printf("\nQueue is empty no elements to delete");return;

}

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

if (data == pri\_que[i]) {

for (; i < rear; i++) {

pri\_que[i] = pri\_que[i + 1];

}

pri\_que[i] = -99;

rear--;

if (rear == -1)

front = -1;

return;

} }

printf("\n%d not found in queue to delete", data);

}

void display\_pqueue() {

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

printf("\nQueue is empty");

return;

}

for (; front <= rear; front++) {

printf(" %d ", pri\_que[front]);

}

front = 0;

}

**Output:**

1 - Insert an element into queue

2 - Delete an element from queue

3 - Display queue elements

4 - Exit

Enter your choice : 1

Enter value to be inserted : 20

Enter your choice : 1

Enter value to be inserted : 30

Enter your choice : 1

Enter value to be inserted : 40

Enter your choice : 3

40 30 20

Enter your choice : 2

Enter value to delete : 40

Enter your choice : 3

30 20

**8.Write a program to implement a queue using stack.**

#include<stdio.h>

#include<stdlib.h>

# define N 20

int s[N],top = -1;

int pop(){

return s[top--];

}

void push(int x){

if(top == N-1)

printf("Stack is Full");

else {

top++;

s[top] = x;

}

}

void enqueue(int x){

push(x);

}

void print(){

int i;

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

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

}

int dequeue(){

int data,res;

if(top == -1)

printf("Queue is Empty");

else if(top == 0)

return pop();

data = pop();

res = dequeue();

push(data);

return res;

}

int main() {

int opt,n,i,data,t;

printf("Enter Your Choice:-");

do{

printf("\n\n1 for Insert the Data in Queue\n2 for show the Data in Queue \n3 for

Delete the data from the Queue\n0 for Exit\n");

scanf("%d",&opt);

switch(opt){

case 1:

printf("\nEnter the number of elements:");

scanf("%d",&n);

printf("\nEnter your data\n");

i=0;

while(i<n){

scanf("%d",&data);

enqueue(data);

i++;

}

break;

case 2:print();

break;

case 3:

t = dequeue();

printf("Dequeued element:%d",t);

break;

case 0:

break;

default:

printf("\nIncorrect Choice");

}

}while(opt!=0);

}

**Output:**

1 for Insert the Data in Queue

2 for show the Data in Queue

3 for Delete the data from the Queue

0 for Exit

1

Enter the number of elements:3

Enter your data

1

2

3

1 for Insert the Data in Queue

2 for show the Data in Queue

3 for Delete the data from the Queue

0 for Exit

3

Dequeued element:1

1 for Insert the Data in Queue

2 for show the Data in Queue

3 for Delete the data from the Queue

0 for Exit

2

3

1 for Insert the Data in Queue

2 for show the Data in Queue

3 for Delete the data from the Queue

0 for Exit

0

**9.Write a program to reverse a queue.**

#include<stdio.h>

int f = -1,r = -1;

int q[50];

void enqueue(int data,int l){

if(r==l-1){

printf("Queue is full");

}

else if((f==-1)&&(r==-1)){

f = r = 0;

q[r] = data;

}

else {

r++;

q[r] = data;

}

}

void print() {

int i;

for(i=f;i<=r;i++) {

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

}

}

void reverse(){

int i,j,t;

for(i=f,j=r;i<j;i++,j--){

t = q[i];

q[i] = q[j];

q[j] = t;}

}

void main() {

int n,i=0,t;

printf("Enter the size of Queue");

scanf("%d",&n);

printf("\nEnter the data for Queue");

while(i<n){

scanf("%d",&t);

enqueue(t,n);

i++;

}

printf("\nQueue which you have entered:-");

print();

reverse();

printf("\nQueue after reversing:-");

print();

}

**Output:**

Enter the size of Queue 5

Enter the data for Queue

6

7

3

2

8

Queue which you have entered:-

6

7

3

2

8

Queue after reversing:-

8

2

3

7

6