Name –Ayan Sadhukhan

Registration Number- 19BEC1448

*Question 1:* Waiting line simulation in post office : In a post office , a line of workers serves a single queue of customers . Every customer receives a token #( serial number) as soon as he/she enters the queue. After the service , the token is returned to the postal worker and the customer leaves the queue. At any point of time the worker may want to know how many customers are yet to be served.

1. Implement the system using an appropriate queue data structure, simulating arrival and departure of customers after service completion.
2. If a customer arrives to operate his/her savings account at the post office, then he/she has attended to first by permitting him/ her to join a special queue. In such a case the postal worker attends to them immediately before resuming his/her normal service . Modify the system to implement this addition in service.

*Code:*

#include<stdio.h>

#define SIZE 5

void enQueue(int);

void enter(int);

void deQueue();

void remove();

void display();

void disp();

int a[SIZE],b[SIZE] ,f1 = -1, r1 = -1,f2=-1,r2=-1;

int main()

{

//deQueue is not possible on empty queue

deQueue();

int ch=0,n=0;

printf("\n 1. Enter the token number \n 2.Enter the token for teachers \n 3.Service \n 4. Print the standing queue\n 5.enter 11 to exit the process \n");

int k=0;

while(ch!=11)

{

printf("\nEnter your choice :");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter the token number");

scanf("%d",&n);

enQueue(n);

break;

case 2:

printf("Inputting number for special faculty \n");

enter(k);

break;

case 3:

if (f2>=0)

remove();

else

deQueue();

case 4:

printf("The queue for the students is \n");

display();

printf("The queue for the faculties is \n");

disp();

break;

default:

printf("\nWrong input\n");

break;

}

}

return 0;

}

void enQueue(int value){

if(r1 == SIZE-1)

printf("\nQueue is Full!!\n");

else {

if(f1 == -1)

f1 = 0;

r1++;

a[r1] = value;

printf("\nInserted -> %d", value);

}

}

void enter(int value){

if(r2 == SIZE-1)

printf("\nQueue is Full!!");

else {

if(f2 == -1)

f2 = 0;

r2++;

b[r2] = value;

printf("\nInserted -> %d", value);

}

}

void deQueue(){

if(f1 == -1)

printf("\nThe Queue for normal people is Empty!!\n");

else{

printf("\nDeleted : %d\n", a[f1]);

f1++;

if(f1 > r1)

f1 = r1 = -1;

}

}

void remove(){

if(f2 == -1)

printf("\nThe queue for the teachers is Empty!!\n");

else{

printf("\nDeleted : %d\n", b[f2]);

f2++;

if(f2 > r2)

f2 = r2 = -1;

}

}

void display(){

if(r1 == -1)

printf("\nQueue is Empty!!!\n");

else{

int i;

printf("\nThe queue is :\n");

for(i=f1; i<=r1; i++)

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

}

}

void disp(){

if(r2 == -1)

printf("\nQueue is Empty!!!\n");

else{

int i;

printf("\nThe faculty queue is :\n");

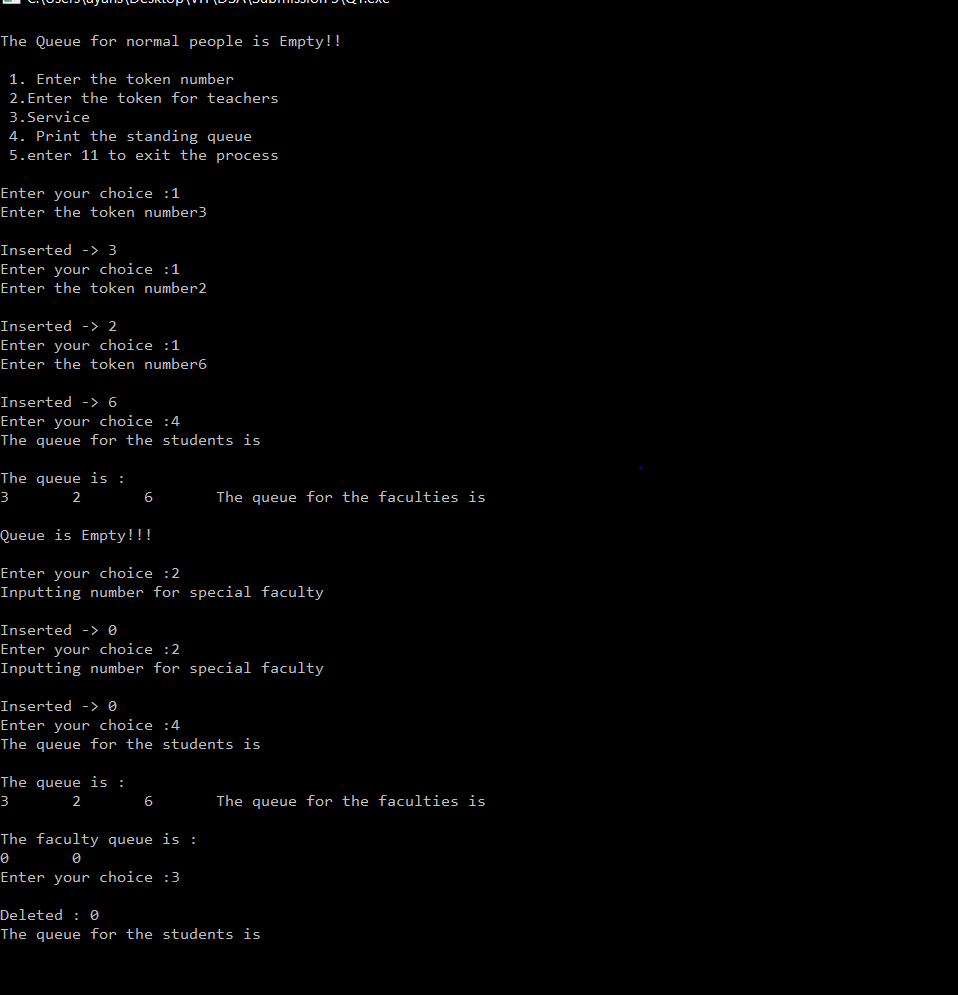
for(i=f2; i<=r2; i++)

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

}

}

*Output:*



*Question 2:* A **Backup\_Queue** is defined as follows: Let Q be a queue of size k. After every k insert operations, we make a copy of the queue for backup purposes. Write a C program to insert ‘n’ operations on the **Backup\_Queue** including copying the queue.

*Code:*

#include<stdio.h>

#define SIZE 5

void enQueue(int);

void deQueue();

void display();

void disp();

void Backup\_Queue();

int items[SIZE],b[100], front = -1, rear = -1,f2=-1,r2=-1;

int main()

{

int ch=0,n=0;

deQueue();

printf("1.Enter the element to the queue \n 2. Delete an element from the queue \n 3. Display the queues \n 4. press 11 to exit from the process \n");

while(ch!=11)

{

printf("\n Enter your choice :");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter the number to inputted in the queue \n");

scanf("%d",&n);

enQueue(n);

break;

case 2:

deQueue();

break;

case 3:

printf("\nThe original queue at the moment is \n");

display();

printf("\nThe backup queue at the moment is \n");

disp();

break;

default:

printf("\nWrong Input\n");

break;

}}

return 0;

}

void enQueue(int value){

if(rear == SIZE-1)

{

printf("\nQueue is Full!!\n");

printf("Performing backup \n");

Backup\_Queue();

rear=front=-1;

enQueue(value);

}

else {

if(front == -1)

front = 0;

rear++;

items[rear] = value;

printf("\nInserted -> %d", value);

}

}

void deQueue(){

if(front == -1)

printf("\nQueue is Empty!!\n");

else{

printf("\nDeleted : %d\n", items[front]);

front++;

if(front > rear)

front = rear = -1;

}

}

void display(){

if(rear == -1)

printf("\n Original Queue is Empty!!!");

else{

int i;

printf("\n Original Queue elements are:\n");

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

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

}

}

void disp(){

if(r2 == -1)

printf("\n Backup Queue is Empty!!!");

else{

int i;

printf("\n Backup Queue elements are:\n");

for(i=f2; i<=r2; i++)

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

}

}

void Backup\_Queue()

{

if(r2 == 99)

{

printf("\n Backup Queue is Full!!\n");

}

else {

int i;

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

{

if(f2 == -1)

f2 = 0;

r2++;

b[r2]=items[i];

}

}

}

*Output:*

