Harsh Kasliwal
2003085
(2)

DATE

Experiment 4.

AIM: Implement Linear Queue ADT using array.

Theory:

Queue ADT: Queue data structure is a collection of similar data items in which insertion and deletion operations are performed based on FIFO principle.

1. enqueue(): Adds a new item to the rear of the queue. It needs the item and returns nothing.

```
Eg Enqueue(val){
      if (isFull()){
          return.}
      else if (isEmpty())
         front=rear=0
      else  rear=rear+1
         queue[rear]=val
```

2. dequeue(): Removes the front item from the queue. It needs no parameters and returns the item.

```
 dequeue(){
   if (isEmpty ()){
          return}
     else
          val = queue[front]
     if (front==rear){
          front=rear=-1}
    else  front=front+1
     return val. }
```

Harsh Kasliwal
2003085

3. Front() : An operation that returns the value of the front element of the queue without deleting it from the front.

```
get-front(){
    if(isEmpty())
        return.
    else return queue[front].
}.
```

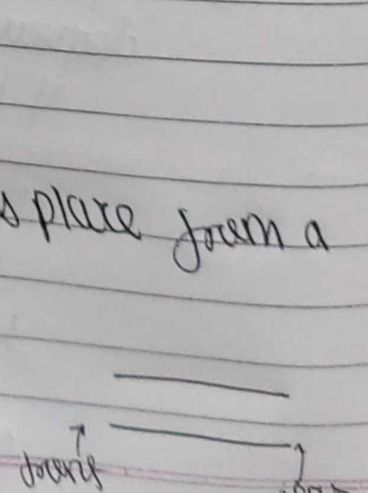
4. isEmpty() : It tests to see whether the queue is empty. It needs no parameters and returns a boolean value.

```
isEmpty(){
    if(front==-1 && rear==-1)
            return True
    else
        return false.
```

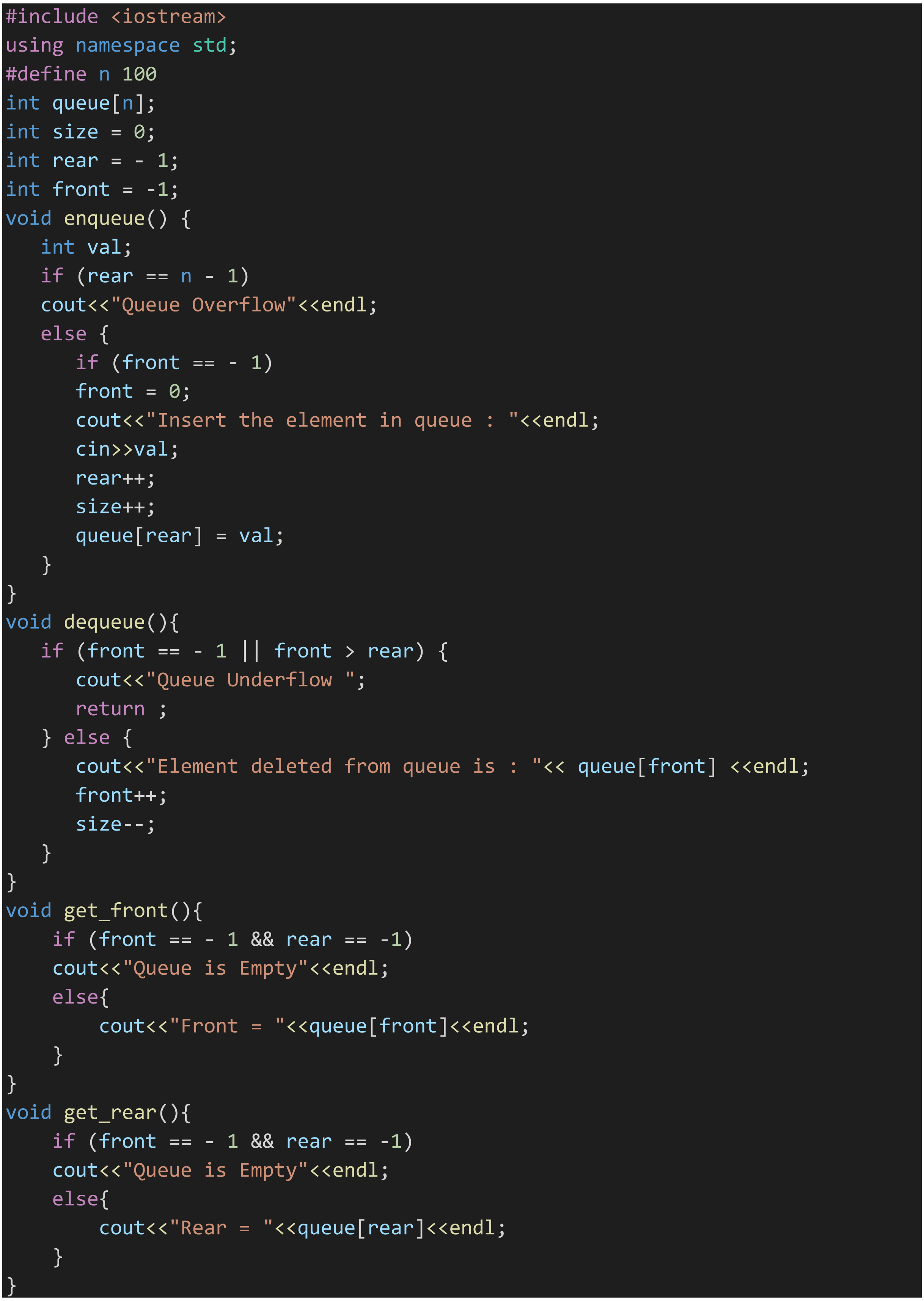
5. isFull() : It tests to see whether the queue is full. It needs no parameters and returns a boolean value.

```
isfull(){
    if(rear==Max-1)
        return True.
    else
        return False.
```

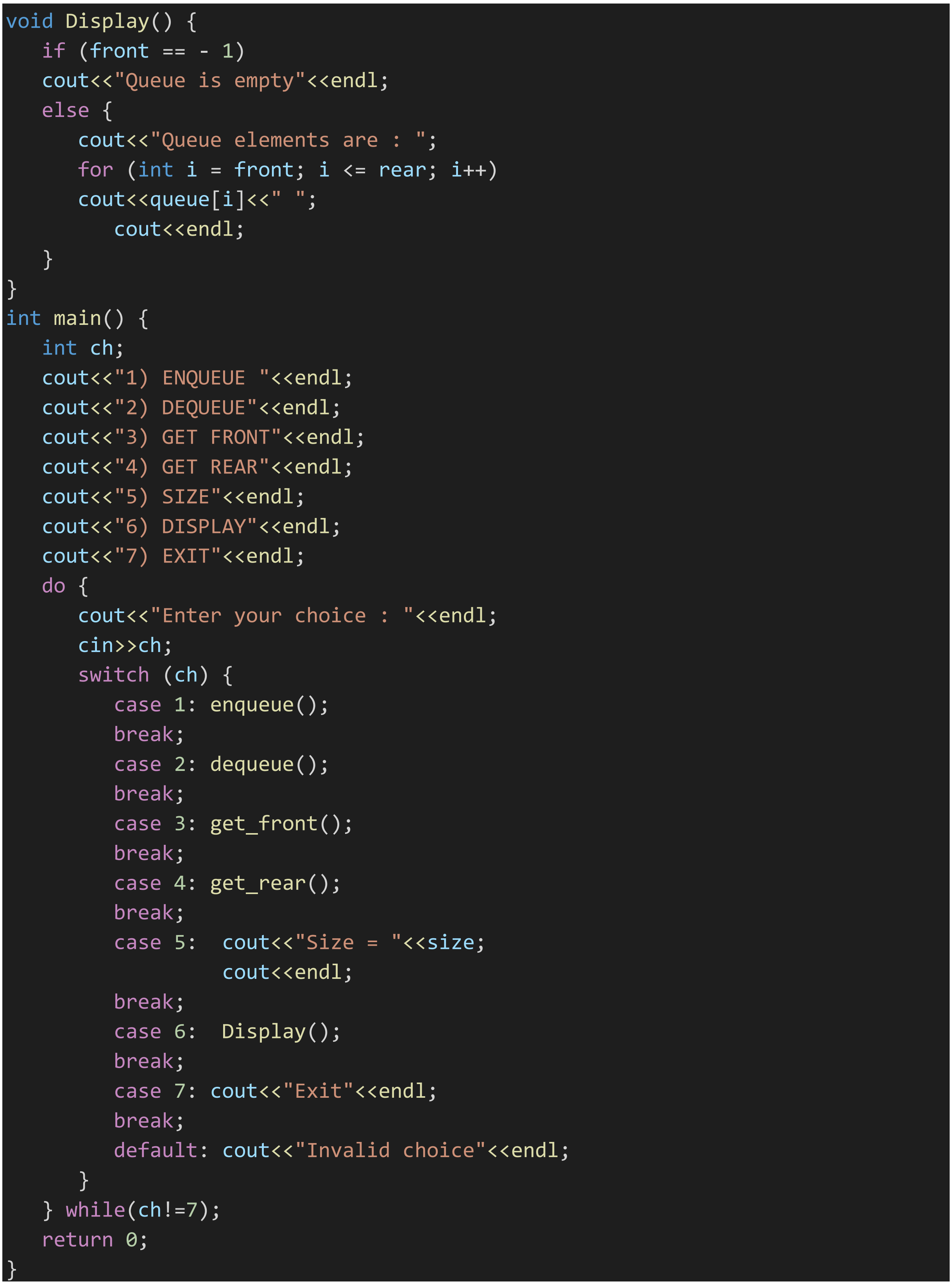
Conclusion : Push and pop operation takes place from a different end of the queue. Based on FIFO (first in first out).



PROGRAM: Write a menu driven code to implement QUEUE ADT using arrays.

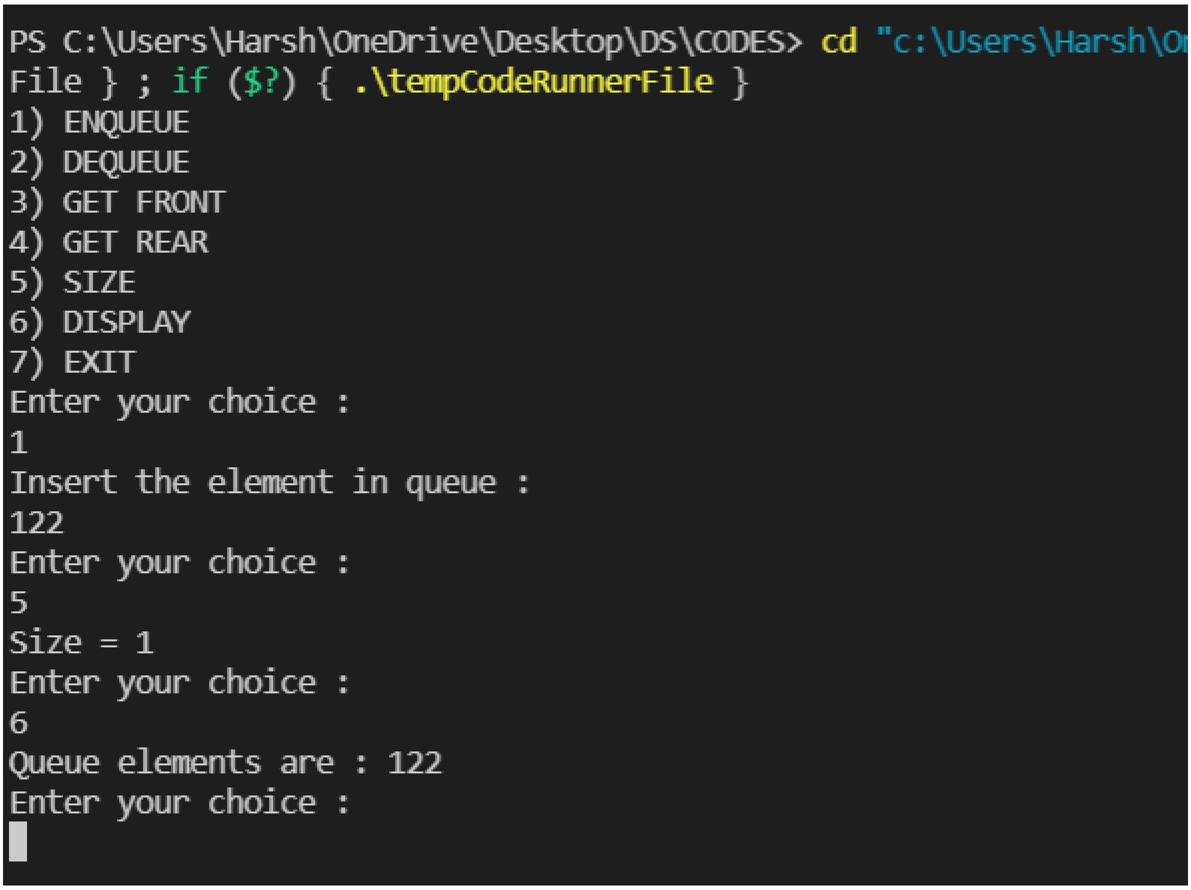


```
#include <iostream>
using namespace std;
#define n 100
int queue[n];
int size = 0;
int rear = - 1;
int front = -1;
void enqueue() {
   int val;
   if (rear == n - 1)
   cout<<"Queue Overflow"<<endl;
   else {
      if (front == - 1)
      front = 0;
      cout<<"Insert the element in queue : "<<endl;
      cin>>val;
      rear++;
      size++;
      queue[rear] = val;
   }
}
void dequeue(){
   if (front == - 1 || front > rear) {
      cout<<"Queue Underflow ";
      return ;
   } else {
      cout<<"Element deleted from queue is : "<< queue[front] <<endl;
      front++;
      size--;
   }
}
void get_front(){
    if (front == - 1 && rear == -1)
    cout<<"Queue is Empty"<<endl;
    else{
        cout<<"Front = "<<queue[front]<<endl;
    }
}
void get_rear(){
    if (front == - 1 && rear == -1)
    cout<<"Queue is Empty"<<endl;
    else{
        cout<<"Rear = "<<queue[rear]<<endl;
    }
}
```



```
void Display() {
   if (front == - 1)
   cout<<"Queue is empty"<<endl;
   else {
      cout<<"Queue elements are : ";
      for (int i = front; i <= rear; i++)
      cout<<queue[i]<<" ";
         cout<<endl;
   }
}
int main() {
   int ch;
   cout<<"1) ENQUEUE "<<endl;
   cout<<"2) DEQUEUE"<<endl;
   cout<<"3) GET FRONT"<<endl;
   cout<<"4) GET REAR"<<endl;
   cout<<"5) SIZE"<<endl;
   cout<<"6) DISPLAY"<<endl;
   cout<<"7) EXIT"<<endl;
   do {
      cout<<"Enter your choice : "<<endl;
      cin>>ch;
      switch (ch) {
         case 1: enqueue();
         break;
         case 2: dequeue();
         break;
         case 3: get_front();
         break;
         case 4: get_rear();
         break;
         case 5:  cout<<"Size = "<<size;
                  cout<<endl;
         break;
         case 6:  Display();
         break;
         case 7: cout<<"Exit"<<endl;
         break;
         default: cout<<"Invalid choice"<<endl;
      }
   } while(ch!=7);
   return 0;
}
```

OUTPUT:



```
PS C:\Users\Harsh\OneDrive\Desktop\DS\CODES> cd "c:\Users\Harsh\O
File } ; if ($?) { .\tempCodeRunnerFile }
1) ENQUEUE
2) DEQUEUE
3) GET FRONT
4) GET REAR
5) SIZE
6) DISPLAY
7) EXIT
Enter your choice :
1
Insert the element in queue :
122
Enter your choice :
5
Size = 1
Enter your choice :
6
Queue elements are : 122
Enter your choice :
```