DATA STRUCTURES LAB

LAB -10

Submitted by

Palash Mishra – 201B172

Submitted to: Dr. KUNJ BIHARI MEENA

2021-2022

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,

AB ROAD, RAGHOGARH, DT. GUNA-473226 MP, INDIA

1. Write a menu driven program to implement linear queue using array and switch-case with following options :

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

```
#include <conio.h>
#include <iostream>
using namespace std;
#define SIZE 10

void enQueue(int);
void deQueue();
void display();

int queue[SIZE], front = -1, rear = -1;

int main()
{
   int value, choice;
   system("cls");
   while (1)
   {
      cout<<"\n\n* MENU *\n";
      cout<<"1. Insertion\n2. Deletion\n3. Front Element\n4. Display\n5. Exit";
      cout<<"\nEnter your choice: ";
      cin>>choice;
      switch (choice)
      {
```

```
        case 1:
            cout<<"Enter the value to be insert: ";
            scanf("%d", &value);
            enQueue(value);
            break;
        case 2:
            deQueue();
            break;
        case 3:
            cout << queue[front];
            break;
        case 4:
            display();
            break;
        case 5:
            exit(0);
        default:
            cout<<"\nInvalid Choice";
        }
    }
}
void enQueue(int value)
{
    if (rear == SIZE - 1)
        cout<<"\nQueue is Full!";
    else
    {
        if (front == -1)
            front = 0;
        rear++;
```

```
        queue[rear] = value;
        cout<<"\nInserted!";
    }
}
void deQueue()
{
    if (front == rear)
        cout<<"\nQueue is Empty";
    else
    {
        cout<<"\nDeleted : %d", queue[front];
        front++;
        if (front == rear)
            front = rear = -1;
    }
}
void display()
{
    if (rear == -1)
        cout<<"\nQueue is Empty!";
    else
    {
        int i;
        cout<<"\nQueue elements are:\n";
        for (i = front; i <= rear; i++)
            cout<<"%d\t", queue[i];
    }
}
```

2. Write a menu driven program to implement circular queue using array and switch-case with following options :

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

[Note: Output Test cases are same as in Que. 1]

```
#include <iostream>
using namespace std;
#define MAX 10

int cqueue_arr[MAX];
int front = -1;
int rear = -1;

void display();
void insert(int item);
int del();
int First_element();
int isEmpty();
int isFull();

int main()
{
   int choice, item;
   while (1)
   {
      cout << "\n1.Insert\n";
      cout << "2.Delete\n";
      cout << "3.First Element\n";
```

```
    cout << "4.Display\n";
    cout << "5.Quit\n";
    cout << "\nEnter your choice : ";
    cin >> choice;

    switch (choice)
    {
    case 1:
       cout << "\nInput the element for insertion : ";
       scanf("%d", &item);
       insert(item);
       break;
    case 2:
       cout << "\nElement deleted is : %d\n"
           << del();
       break;
    case 3:
       cout << "\nElement at the front is  : %d\n"
           << First_element();
       break;
    case 4:
       display();
       break;
    case 5:
       exit(1);
    default:
       cout << "\nWrong choice\n";
    }
}
```

```
    return 0;
}

void insert(int item)
{
    if (isFull())
    {
        cout << "\nQueue Overflow\n";
        return;
    }
    if (front == -1)
        front = 0;

    if (rear == MAX - 1)
        rear = 0;
    else
        rear = rear + 1;
    cqueue_arr[rear] = item;
}

int del()
{
    int item;
    if (isEmpty())
    {
        cout << "\nQueue Underflow\n";
        exit(1);
    }
    item = cqueue_arr[front];
    if (front == rear)
```

```
    {
        front = -1;
        rear = -1;
    }
    else if (front == MAX - 1)
        front = 0;
    else
        front = front + 1;
    return item;
}

int isEmpty()
{
    if (front == -1)
        return 1;
    else
        return 0;
}

int isFull()
{
    if ((front == 0 && rear == MAX - 1) || (front == rear + 1))
        return 1;
    else
        return 0;
}

int First_element()
{
    if (isEmpty())
```

```
    {
        cout << "\nQueue Underflow\n";
        exit(1);
    }
    return cqueue_arr[front];
}

void display()
{
    int i;
    if (isEmpty())
    {
        cout << "\nQueue is empty\n";
        return;
    }
    cout << "\nQueue elements :\n";
    i = front;
    if (front <= rear)
    {
        while (i <= rear)
            cout << cqueue_arr[i++];
    }
    else
    {
        while (i <= MAX - 1)
            cout << cqueue_arr[i++] << " ";
        i = 0;
        while (i <= rear)
            cout << cqueue_arr[i++] << " ";
    }
```

```
    cout << "\n";
}
```

3. Write a menu driven program to implement linear queue using linked list and switch-case with following options :

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

[Note: Output Test cases are same as in Que. 1]

```
#include <iostream>
#include <conio.h>
#define size 20
using namespace std;
struct Node
{
    int data;
    Node *next;
};
struct Node *front = NULL, *rear = NULL;
void enqueue(int val)
{
    struct Node *newNode = new struct Node;
    newNode->data = val;
    newNode->next = NULL;
    if (front == NULL && rear == NULL)
        front = rear = newNode;
    else
    {
```

```
        rear->next = newNode;
        rear = newNode;
    }
}
void dequeue()
{
    struct Node *Temp;
    Temp = front;
    if (front == NULL)
        cout << "Queue Is Empty\n";
    else
    {
        cout << "Dequeued Element is : " << front->data;
        front = front->next;
        if (front == NULL)
            rear = NULL;
        delete (Temp);
    }
}
void display()
{
    struct Node *temp = front;
    while (temp != NULL)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
}
int main()
{
```

```
int ch;
xx:
system("cls");
cout << "1.Insert\n";
cout << "2.Delete\n";
cout << "3.Display element at the front\n";
cout << "4.Display all elements of the queue\n";
cout << "5.Quit\n";
cout<<"Enter Your Choice\n";
cin>>ch;
switch (ch)
{
case 1:
   int value;
   cout<<"\nInput the element for adding in queue : ";
   cin>>value;
   enqueue(value);
   break;
case 2:
   dequeue();
   getch();
   break;
case 3:
   cout<<front->data;
   getch();
   break;
case 4:
   display();
   getch();
   break;
```

```
    case 5:
        exit(0);
    default:
        cout<<"\nOOPS! Enter the correct choice : ";
        break;
    }
    goto xx;
    return 0;
}
```

4. WAP to implement priority queue with its basic operations.

```
#include <iostream>
using namespace std;
#define MAX 5
void insert(int);
void Delete(int);
void create();
void check(int);
void display_pqueue();

int pri_que[MAX];
int front, rear;

int main()
{
    int n, ch;

    cout<<"\n1 - Insert an element into queue";
    cout<<"\n2 - Delete an element from queue";
    cout<<"\n3 - Display queue elements";
    cout<<"\n4 - Exit";
```

```
    create();

    while (1)
    {
       cout("\nEnter your choice : ");
       scanf("%d", &ch);

       switch (ch)
       {
       case 1:
          cout("\nEnter value to be inserted : ");
          scanf("%d",&n);
          insert(n);
          break;
       case 2:
          cout("\nEnter value to delete : ");
          scanf("%d",&n);
          Delete(n);
          break;
       case 3:
          display_pqueue();
          break;
       case 4:
          exit(0);
       default:
          cout("\nChoice is incorrect");
       }
    }
}
```

```
void create()
{
   front = rear = -1;
}
void insert(int data)
{
   if (rear >= MAX - 1)
   {
      cout("\nQueue overflow");
      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(int data)
{
   int i;

   if ((front==-1) && (rear==-1))
   {
      cout("\nQueue is empty");
      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;
        }
    }
    cout("\n%d not found in queue to delete", data);
}
void display_pqueue()
{
    if ((front == -1) && (rear == -1))
    {
        cout("\nQueue is empty");
        return;
    }

    for (; front <= rear; front++)
    {
        cout(" %d ", pri_que[front]);
    }

    front = 0;
}
```