# CL-218

**Data Structures Lab # 3**

**Objectives:**

1. Queue Operations
2. Priority Queue

**Note: Carefully read the following instructions (***Each instruction contains a weightage***)**

1. There must be a block of comments at start of every question's code; the block should contain brief description about functionality of code
2. Proper indentation of code is essential
3. Variable name should be meaningful
4. Make a Microsoft Word file and past all of your C++ code with screenshot of outputs in MS word.
5. First think about statement problems and then write/draw your logic on copy.
6. After copy pencil work, code the problem statement on MS Studio C++ compiler.
7. At the end when you done your today lab tasks, attached only MS word file and make your submission on Google Classroom.
8. Late and email submission is not accepted. All tasks must be submitted during the lab time

**GitHub Link:** <https://github.com/atariq12382/Data-Structures-Lab.git>

**Problem 1:** **Array base queue implementation**

Provide array base implementation of queue. Also create a driver function

**Functions:**

1. Enqueue()
2. Dequeue()
3. Isempty()
4. Isfull()

**Solution:**

#include<iostream>

using namespace std;

class Queue {

private:

int front;

int rear;

int arr[5];

public:

Queue() {

front = -1;

rear = -1;

for (int i = 0; i < 5; i++) {

arr[i] = 0;

}

}

bool isEmpty() {

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

return true;

else

return false;

}

bool isFull() {

if (rear == 4)

return true;

else

return false;

}

void enqueue(int val) {

if (isFull()) {

cout << "Queue full" << endl;

return;

}

else if (isEmpty()) {

rear = 0;

front = 0;

arr[rear] = val;

}

else {

rear++;

arr[rear] = val;

}

}

int dequeue() {

int x = 0;

if (isEmpty()) {

cout << "Queue is Empty" << endl;

return x;

}

else if (rear == front) {

x = arr[rear];

rear = -1;

front = -1;

return x;

}

else {

cout << "front value: " << front << endl;

x = arr[front];

arr[front] = 0;

front++;

return x;

}

}

void display() {

cout << "All values in the Queue are - " << endl;

for (int i = 0; i < 5; i++) {

cout << arr[i] << " ";

}

}

};

int main() {

Queue q1;

int value, option;

bool condition = 1;

while(condition)

{

system("cls");

cout << "Please selectfrom the following:" << endl;

cout << "1. Enqueue()" << endl;

cout << "2. Dequeue()" << endl;

cout << "3. isEmpty()" << endl;

cout << "4. isFull()" << endl;

cout << "5. display()" << endl;

cout << "pleas enter the number of selected option:";

cin >> option;

system("cls");

switch (option) {

case 1:

cout << "Enter a number you want to enqueue:" << endl;

cin >> value;

q1.enqueue(value);

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 2:

cout << "Dequeued Value : " << q1.dequeue() << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 3:

if (q1.isEmpty())

cout << "Queue is Empty" << endl;

else

cout << "Queue is not Empty" << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 4:

if (q1.isFull())

cout << "Queue is Full" << endl;

else

cout << "Queue is not Full" << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 5:

cout << "Queue is:-" << endl;

q1.display();

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

default:

cout << "Enter Proper Option number." << endl;

cout << "If you want to perform any other function press 1 else press 0";

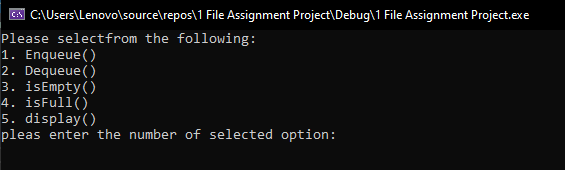
cin >> condition;

break;

}

}

}



**Problem 2:**

Provide Linked list base implementation of queue. Also create a driver function

**Functions:**

1. Enqueue()

1. Dequeue()
2. Isempty()
3. Isfull()

**Solution:**

#include<iostream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

Node()

{

data = 0;

next = NULL;

}

};

class List

{

public:

Node\* head;

Node\* temp;

List()

{

head = NULL;

temp = NULL;

}

bool isEmpty()

{

if (head == NULL)

return 1;

else

return 0;

}

bool isFull()

{

Node\* temp;

try

{

temp = new Node;

delete temp;

return 0;

}

catch (bad\_alloc exception)

{

return 1;

}

}

void append(int a)

{

Node\* ptr;

if (isFull())

{

cout << "Stack is Full" << endl;

return;

}

else

{

if (isEmpty())

{

head = new Node;

head->data = a;

}

else

{

temp = new Node;

ptr = head;

while (ptr->next != NULL)

{

ptr = ptr->next;

}

temp->data = a;

ptr->next = temp;

}

}

}

int nodeValue()

{

int val;

Node\* del;

del = head;

val = del->data;

head = del->next;

delete(del);

return val;

}

Node\* returnHead()

{

return head;

}

Node\* returnTemp()

{

return temp;

}

};

class Queue

{

public:

List l;

Node\* front;

Node\* rear;

Queue()

{

front = rear = NULL;

}

bool isFull()

{

bool re;

re = l.isFull();

return re;

}

bool isEmpty()

{

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

return 1;

else

return 0;

}

void enqueue(int a)

{

if (isEmpty())

{

l.append(a);

front = l.returnHead();

rear = l.returnHead();

}

else

{

l.append(a);

rear = l.returnTemp();

}

}

int dequeue()

{

int val;

if (isEmpty())

{

cout << "Stack is Empty" << endl;

return -1111;

}

else

{

val = l.nodeValue();

front = l.returnHead();

if (front == NULL)

rear = NULL;

return val;

}

}

};

int main()

{

Queue q1;

int value, option;

bool condition = 1;

while (condition)

{

system("cls");

cout << "Please selectfrom the following:" << endl;

cout << "1. Enqueue()" << endl;

cout << "2. Dequeue()" << endl;

cout << "3. isEmpty()" << endl;

cout << "4. isFull()" << endl;

cout << "pleas enter the number of selected option:";

cin >> option;

system("cls");

switch (option) {

case 1:

cout << "Enter a number you want to enqueue:";

cin >> value;

q1.enqueue(value);

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 2:

cout << "Dequeued Value : " << q1.dequeue() << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 3:

if (q1.isEmpty())

cout << "Queue is Empty" << endl;

else

cout << "Queue is not Empty" << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 4:

if (q1.isFull())

cout << "Queue is Full" << endl;

else

cout << "Queue is not Full" << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

default:

cout << "Enter Proper Option number." << endl;

cout << "If you want to perform any other function press 1 else press 0";

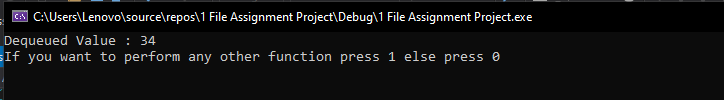
cin >> condition;

break;

}

}

}



**Problem 3:**

Create priority queue using linked lists. Give the each node a priority number to represent its priority. Most prior element would have lowest the value numeric of priority

#include<iostream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

Node()

{

data = 0;

next = NULL;

}

};

class List

{

public:

Node\* head;

Node\* temp;

List()

{

head = NULL;

temp = NULL;

}

bool isEmpty()

{

if (head == NULL)

return 1;

else

return 0;

}

bool isFull()

{

Node\* temp;

try

{

temp = new Node;

delete temp;

return 0;

}

catch (bad\_alloc exception)

{

return 1;

}

}

void append(int a)

{

Node\* ptr;

if (isFull())

{

cout << "Stack is Full" << endl;

return;

}

else

{

if (isEmpty())

{

head = new Node;

head->data = a;

}

else

{

temp = new Node;

ptr = head;

while (ptr->next != NULL)

{

ptr = ptr->next;

}

temp->data = a;

ptr->next = temp;

}

}

}

int nodeValue()

{

int val;

Node\* del;

del = head;

val = del->data;

head = del->next;

delete(del);

return val;

}

Node\* returnHead()

{

return head;

}

Node\* returnTemp()

{

return temp;

}

void bubbleSort()

{

Node\* ptr;

Node\* ptr1;

Node\* ptr2;

Node\* swap;

Node\* swap2;

Node\* del;

Node\* min;

ptr = head;

while (ptr != NULL)

{

ptr1 = head;

ptr2 = ptr1->next;

while (ptr2 != NULL)

{

min = ptr1;

if (ptr1->data > ptr2->data)

min = ptr2;

if (min == ptr1)

{

ptr1 = ptr2;

ptr2 = ptr2->next;

continue;

}

if (ptr1 == head)

{

head = min;

swap = ptr1;

}

else

{

swap = head;

while (swap->next != min)

{

swap = swap->next;

if (swap == NULL)

break;

}

}

if (swap != ptr1)

{

del = ptr1->next;

ptr1->next = ptr2->next;

ptr2->next = ptr1;

swap->next = ptr2;

ptr1 = ptr2->next;

ptr2 = ptr1->next;

}

else

{

if (min == head)

{

ptr1->next = ptr2->next;

ptr2->next = ptr1;

ptr1 = ptr2->next;

ptr2 = ptr1->next;

}

else

{

swap2 = head;

while (swap2->next != ptr1)

{

swap2 = swap2->next;

if (swap2 == NULL)

break;

}

del = ptr1->next;

ptr1->next = ptr2->next;

swap2->next = ptr2;

ptr2->next = ptr1;

ptr1 = ptr2->next;

ptr2 = ptr1->next;

}

}

}

ptr = ptr->next;

}

}

};

class Queue

{

public:

List l;

Node\* front;

Node\* rear;

Queue()

{

front = rear = NULL;

}

bool isFull()

{

bool re;

re = l.isFull();

return re;

}

bool isEmpty()

{

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

return 1;

else

return 0;

}

void enqueue(int a)

{

if (isEmpty())

{

l.append(a);

front = l.returnHead();

rear = l.returnHead();

}

else

{

l.append(a);

rear = l.returnTemp();

l.bubbleSort();

}

}

int dequeue()

{

int val;

if (isEmpty())

{

cout << "Stack is Empty" << endl;

return -1111;

}

else

{

val = l.nodeValue();

front = l.returnHead();

if (front == NULL)

rear = NULL;

return val;

}

}

};

int main()

{

Queue q1;

int value, option;

bool condition = 1;

while (condition)

{

system("cls");

cout << "Please selectfrom the following:" << endl;

cout << "1. Enqueue()" << endl;

cout << "2. Dequeue()" << endl;

cout << "3. isEmpty()" << endl;

cout << "4. isFull()" << endl;

cout << "pleas enter the number of selected option:";

cin >> option;

system("cls");

switch (option) {

case 1:

cout << "Enter a number you want to enqueue:";

cin >> value;

q1.enqueue(value);

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 2:

cout << "Dequeued Value : " << q1.dequeue() << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 3:

if (q1.isEmpty())

cout << "Queue is Empty" << endl;

else

cout << "Queue is not Empty" << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

case 4:

if (q1.isFull())

cout << "Queue is Full" << endl;

else

cout << "Queue is not Full" << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

default:

cout << "Enter Proper Option number." << endl;

cout << "If you want to perform any other function press 1 else press 0";

cin >> condition;

break;

}

}

}

