**DATA STRUCTURES LAB 8**

NAME: Ahmed Kasteer

ROLL NO: 20F-0336

SECTION 3D

QUESTION 1

#include <iostream>

using namespace std;

struct node

{

int data;

node\* next;

node(int value)

{

data = value;

next = NULL;

}

};

class Queue

{

public:

node\* front;

node\* rear;

int itemNums;

Queue()

{

front = NULL;

rear = NULL;

itemNums = 0;

}

void Enqueue(int x)

{

node\* temp = new node(x); // create a new node

if (rear == NULL) // if queue is empty

{

front = rear = temp;

return;

}

rear->next = temp; //inserting new node

rear = temp;

itemNums++;

}

void Dequeue()

{

if (front == NULL) // if queue is empty

{

cout << "Queue is already empty " << endl;

return;

}

node\* temp = front; // storing the value at front and then moving the front ahead

front = front->next;

if (front == NULL) // if front becomes NULL, rear also becomes NULL

rear = NULL;

delete (temp);

}

void duplicate(int x)

{

for (int i = 0; i < x; i++)

{

Enqueue(x);

}

}

void Print()

{

if (front == NULL)

{

cout << "The queue is empty" << endl;

return;

}

else

{

node\* temp = front;

while (temp != NULL)

{

cout << temp->data << " ";

temp = temp->next;

}

}

}

};

int main()

{

Queue q;

q.duplicate(2);

q.duplicate(3);

q.duplicate(4);

q.Print();

}

Text

Description automatically generated

Question 2

#include <iostream>

using namespace std;

class Queue

{

public:

int\* qArray;

int size;

int front;

int rear;

int itemNums;

Queue()

{

size = 10;

qArray = new int[size];

front = -1;

rear = -1;

itemNums = 0;

}

bool IsEmpty()

{

if (itemNums == 0)

return true;

else

return false;

}

bool IsFull()

{

if (itemNums == size)

return true;

else

return false;

}

void Enqueue(int value)

{

if (IsFull() == true)

{

cout << "The queue is full." << endl;

return;

}

rear = (rear + 1) % size;

qArray[rear] = value;

itemNums++;

}

void Display()

{

cout << "Priority queue: ";

if (IsEmpty() == true)

{

cout << "The queue is empty." << endl;

}

else

{

for (int i = 0; i < 4; i++)

{

cout << qArray[i] << " ";

}

}

}

};

int main()

{

int items[4] = { 0 };

int priority;

int number;

Queue queue;

for (int i = 0; i < 4; i++)

{

cout << "Enter Number: ";

cin >> number;

cout << "Enter priority: ";

cin >> priority;

items[priority - 1] = number;

cout << endl;

}

for (int i = 0; i < 4; i++)

{

queue.Enqueue(items[i]);

}

queue.Display();

}

Text

Description automatically generated

Question 3

#include <iostream>

using namespace std;

struct node

{

string message;

node\* next;

int priority;

node(string value)

{

message = value;

next = NULL;

priority = 0;

}

};

class Queue

{

public:

node\* front;

node\* rear;

Queue()

{

front = NULL;

rear = NULL;

}

void Enqueue(string value, int pri)

{

node\* temp = new node(value);

if (front == NULL)

{

front = rear = temp;

}

else if (pri == 0)

{

temp->next = front;

front = temp;

}

else if (pri >= 1)

{

node\* curr = front;

int i = 0;

while (i != pri)

{

if (i == pri)

{

temp->next = curr->next;

curr->next = temp;

break;

}

curr = curr->next;

i++;

}

}

}

void display()

{

int i = 1;

for (node\* temp = front; temp != NULL; temp = temp->next)

{

cout << "Message " << i << ": " << temp->message << endl;

i++;

}

}

};

int main()

{

Queue q;

cout << "Buffered messages displaying according to priority of them after being online: " << endl;

//q.Enqueue(message, priority)

q.Enqueue("Hello", 2);

q.Enqueue("Hope you are fine.", 0);

q.display();

}

Text

Description automatically generated

Question 4

#include<iostream>

using namespace std;

class Queue

{

public:

char\* QueueArray;

int front;

int rear;

int itemNums;

int arraysize;

int\* pt; //process time

char proc; //process

Queue(int s)

{

arraysize = s;

front = -1;

rear = -1;

itemNums = 0;

pt = new int[arraysize];

QueueArray = new char[arraysize];

proc = ' ';

for (int i = 0; i < arraysize; i++)

{

QueueArray[i] = ' ';

}

}

bool IsEmpty()

{

if (front == -1)

return true;

else

return false;

}

bool IsFull()

{

if (rear == arraysize - 1)

return true;

else

return false;

}

void Enqueue()

{

if (IsFull() == true)

cout << "The queue is full." << endl;

else

{

if ((front == -1) && (rear == -1)) // checking if first element is being entered

{

cout << "Enter Process: ";

cin >> proc;

front = 0;

rear = 0;

QueueArray[rear] = proc;

itemNums++;

}

else

{

cout << "Enter Process: ";

cin >> proc;

rear = (rear + 1) % arraysize;

QueueArray[rear] = proc;

itemNums++;

}

cout << "Enter Process time: ";

cin >> pt[rear];

cout << endl;

}

}

void Dequeue()

{

if (IsEmpty() == true)

cout << "The queue is empty." << endl;

else

{

front = (front + 1) % arraysize;

QueueArray[front] = NULL;

itemNums--;

}

}

void sort()

{

for (int i = 0; i < arraysize; i++)

{

if (pt[i] - 2 == 0)

Dequeue();

else

{

Dequeue();

Enqueue();

}

}

}

void Display()

{

for (int i = 0; i < arraysize; i++)

{

cout << QueueArray[i] << endl;

}

}

};

int main()

{

int arraysize;

cout << "Enter number of processes: ";

cin >> arraysize;

cout << endl;

Queue q(arraysize);

for (int i = 0; i < arraysize; i++)

{

q.Enqueue();

}

q.sort();

q.Display();

}