Data Structures Lab 7

Name: Ahmed Kasteer

Rollno: 20F0336

Section 3D

Question 2

#include<iostream>

using namespace std;

void towerOfHanoi(int disknum, char fromRod, char toRod, char auxRod)

{

if (disknum == 1)

{

cout << "Move disk 1 from rod " << fromRod << " to rod " << toRod << endl;

return;

}

towerOfHanoi(disknum - 1, fromRod, auxRod, toRod);

cout << "Move disk " << disknum << " from rod " << fromRod << " to rod " << toRod << endl;

towerOfHanoi(disknum - 1, auxRod, toRod, fromRod);

}

int main()

{

cout << "When n = 2: " << endl;

int n = 4;

towerOfHanoi(n, 'A', 'C', 'B');

cout << endl;

cout << "When n = 3: " << endl;

n = 3;

towerOfHanoi(n, 'A', 'C', 'B');

cout << endl;

cout << "When n = 5: " << endl;

n = 5;

towerOfHanoi(n, 'A', 'C', 'B');

return 0;

}

Text

Description automatically generated

Question 5

#include<iostream>

using namespace std;

class Queue

{

public:

int front, rear, capacity;

int\* queue;

Queue(int c)

{

front = rear = 0;

capacity = c;

queue = new int;

}

void Enqueue(int data)

{

if (capacity == rear)

{

cout << "Queue is full." << endl;

return;

}

else

{

queue[rear] = data;

rear++;

}

return;

}

void Dequeue()

{

if (isEmpty() == true)

{

return;

}

else

{

for (int i = 0; i < rear - 1; i++)

{

queue[i] = queue[i + 1];

}

rear--;

}

return;

}

bool isEmpty()

{

if (front == rear)

{

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

return true;

}

else

{

return false;

}

}

void Display()

{

if (front == rear) {

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

return;

}

cout << endl;

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

{

cout << queue[i] << " ";

}

return;

}

};

int main(void)

{

Queue q(4);

cout << "Queue: " << endl;

q.Display();

q.Enqueue(20);

q.Enqueue(30);

q.Enqueue(40);

q.Enqueue(50);

q.Display();

cout << endl;

cout << "When another node was inserted: " << endl;

q.Enqueue(60);

q.Display();

q.Dequeue();

q.Dequeue();

cout << endl;

cout << "after two node deletion: " << endl;

q.Display();

}