Question 1

A picture containing clipart

Description automatically generated

A queue maintains a set of data which will be stored in a sequence and is able to store more data from one end of the sequence and can be removed from the other end. To be more specific a queue algorithm follows the FIFO rule, the First in First Out. Which unlike the stack algorithm has the ability to take the data out that was stored first, while in the stack algorithm which follows the LIFO rule where the first stored data will always be the last to be taken out. For the above graph it is not possible to go from level to level. For queue, just like the example in class, in order to go for the root to the next level you would need to dequeue the previous level in order to the get the next set of siblings.

For example: The root for this graph would be 50 – so then the queue would traverse through the level one that is connected to the root:

So, the queue would contain – [2,30,14,60] with root 50 but in order to go to the next level you would need to dequeue the current queue. If the list was dequeued the root would be 2 (since it will follow the FIFO) and the rest would be lost.

Following the pattern now 2 is now the root and the queue contains – [15,25] and following the pattern again 15 would become the root and 25 would be removed, but after checking for the child of the now root 15, there would no children and the program would end. How could the program recover the dequeued data? Unless of course if you have saved it somewhere. Like the 2 assignments before where we used the stack algorithm to solve a order of operations problem and the previously solved nodes were saved in the stack itself. With the limits of the question (and the lack of detail) I have come to the conclusion that it is not possible to be using the stack.

Even of there was a way to use the queue algorithm it would take longer, takes more memory so it would be illogical and inefficient to use a queue to traverse through this graph. Now queue would be effective when solving an undirected graph (as was seen in the last assignment) even a directed graph.

Question 2

A way to implement this type of problem is by doing this(as per my understanding and notes references so far):

For ***push*** lets input the 1st data, then 2nd and the pattern follow. Next enqueue the element in the second queue and transfer over a copy of the elements in the 1st queue into the 2nd queue.

For ***pop*** is fairly simple you would just dequeue the data from the queue from the last inputted data item.

public void push (int x) {

If (q1 is empty) {

q1 and enqueue(x)

} else {q2 and enqueue (x);

while (q1 not empty) q2 enqueue transfers (q2 when dequeues)

if not empty

while (q2 not empty) q1 enqueue transfers (q1 when dequeues)

}}

public int pop () {

int pData = q2 dequeues

return pData;

}