Lecture 4: Linked List and Stacks/Queues

Class	Cohort 2 Year 1
Created	@Oct 14, 2020 2:47 PM
Materials	
Reviewed	☐
Type	Lecture

Linked List Questions

1. We need to insert a new object at the end of the linked list.
2. Search in singly linked list
3. Deletion by value
4. Find the length of a linked list
5. Reverse a linked list
6. Detect loop in a linked list
7. Find Middle Node of linked list
8. You will now be implementing the removeDuplicates() function. When a linked list is passed to this function, it removes any node which is a duplicate of another existing node.
9. Union and Intersection
10. Return the Nth node from end

Stacks

Implementation

```
#include "stack.h"
#include <cassert>
```

```
myStack::myStack(int size) {
  capacity = size;
  stackArr = new int[size];
  assert(stackArr != NULL);
  numElements = 0;
}

bool myStack::isEmpty() {
  return (numElements == 0);
}

int myStack::getTop() {
  return (numElements == 0 ? -1 : stackArr[numElements - 1]);
}

bool myStack::push(int value) {
  if (numElements < capacity) {
    stackArr[numElements] = value;
    numElements++;
    return true;
  } else {
    cout << "Stack Full." << endl;
    return false;
  }
}

int myStack::pop() {
  if (numElements == 0) {
    cout << "Stack Empty" << endl;
    return -1;
  } else {
    numElements--;
    return stackArr[numElements];
  }
}

int myStack::getSize() {
  return numElements;
}

void myStack::showStack() {
  int i = 0;
  while (i < numElements) {
    cout << '\t' << stackArr[numElements - 1 - i];
    i++;
  }
  cout << '\n';
}
```

Queue

Implementation

```
#include "queue.h"
#include <cassert>

using namespace std;

myQueue::myQueue(int size) {
  capacity = size;
  queueArr = new int[size];
  assert(queueArr != NULL);
  numElements = 0;
  front = 0;
  back = -1;

}

bool myQueue::isEmpty() {
  return (numElements == 0);
}

int myQueue::getFront() {
  if (isEmpty()) {
    cout << "Queue Empty" << endl;
    return -1;
  } else
    return queueArr[front];
}

void myQueue::enqueue(int value) {
  if (numElements == capacity) {
    cout << "Queue Full" << endl;
    return;
  }

  if (back == capacity - 1)
    back = -1;

  queueArr[++back] = value;
  numElements++;
}

int myQueue::dequeue() {
  if (isEmpty()) {
    cout << "Queue Empty" << endl;
    return -1;
  }
  int tmp = queueArr[front++];

  if (front == capacity)
    front = 0;
  numElements--;
  return tmp;

}
int myQueue::getSize() {
  return numElements;
}
```

```
void myQueue::showqueue() {
  int i = front;
  int count = 0;
  while (count != numElements) {
    cout << '\t' << queueArr[i%capacity];
    i++;
    count++;
  }
  cout << '\n';
}
```