PROG23672-Trafalgar-Winter2018-Assignment 2

Due date: Feb 14, 2020 11:59 pm

Submit to SLATE at DROPBOX

Part I (60 marks): Demonstrating how mathematical induction improves algorithm efficiency.

(a) **(10 marks):** Write an algorithm in pseudo code to calculate the sum of the first n even numbers. The algorithm receives two inputs: (1) an array of the first n even numbers called a; (2) the length of the array called n. The algorithm must use one loop at least.

(b) **(20 marks):** Prove by induction that the sum of the first n even numbers is n (n+1). In mathematical language, we can describe the i^{th} even number by 2i. This description allows us to cast the problem as follows:

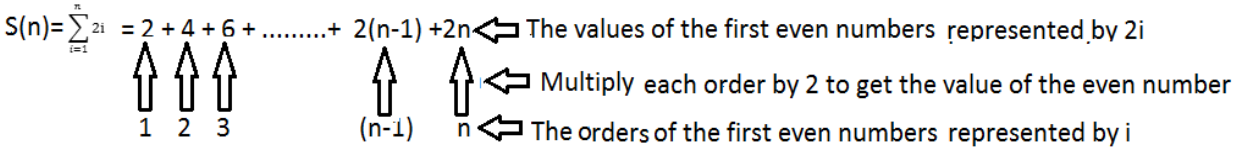
$$\text{Summation of first n even numbers} = \mathrm{S(n)} = \sum_{i=1}^{n} 2\mathrm{i} = n(n+1)$$

Where:

i is an integer that represents the order of an even number in the first n even numbers

n is the count of the numbers in an even number set that starts with the value 2 when i=1 and ends with the value 2n when i=n.

The summation of the first n even number can be clarified by the following:



(c) **(10 marks):** Write an algorithm in pseudo code to calculate the sum of the first n even numbers. The algorithm receives the count of the first even numbers to be summed. It uses the theorem presented in b s(n)= n(n+1) to calculate the summation

(d) **(20 marks):** Compare the two algorithms developed in (a) and (c) by estimating their efficiencies (Big Oh) and clarifying the following:

- The number of primitive operations for each algorithm when n=10, n=100, n=10000, and n=1000000.
- Why is the mathematical induction in (b) is so important?

For part 1: You can use Microsoft word or any word-processing software to write your answers. If you want to use hand writing to answer part 1, you must scan your answers and submit them to SLATE.

Part 2 (40 marks): Implementing your own generic linked list ADT

In this part you will construct and use your own reusable implementation of a generic linked list ADT. The purpose is to develop a reusable generic linked list class and use it.

The list interface: Your Linked list should support the following Linked List interface:

```
template <typename E>
class SLinkedList {                // a singly linked list
public:
       SLinkedList();              // empty list constructor
       ~SLinkedList();             // destructor
       bool empty() const;         // is list empty? Returns true or false
       const E& front() const;     // returns front element
       const E& last() const;      // return last element
       void addFront(const E& e);  // add e to front of list
       void addLast(const E& e);   // add e to rear of list
       void removeFront();         // remove the front element from list
       void removeLast();          // remove the last element from list
       void print();               // print the list
private:
       SNode<E>* head;             // head of the list - a pointer to the first node
       SNode<E>* tail;             // tail of the list - apointer to the last node

};
```

The list functions (30 marks): You should create a MyLinkedList class (MyLinkedList.cpp) that implements the SLinkedList interface shown above. You can use the SlinkedList code provided in the class and add to it the implementations of new four functions: last(), addLast(), removeLast(), and print()

The last () function returns the last element in the linked list. The function Addlast() creates a new element and attaches it to the end of the list. The function removeLast() removes the last element from the list and releases its memory. The print() function should simply output every element the list contains in the list order, with a space between each element. The print () function should leave its list in the same state it begins with.

Be sure to take into account that one might try to remove a node from an empty list. Put (or download from SLATE) the following exception in the file `MyExceptions.h`. Then you can `throw new linkedlistException(" empty linked list")`, for example.

```
class LinkedException : public RuntimeException {
public:
       LinkedListException(const string& err) : RuntimeException(err) {}
};
```

The print function also throws a LinkedListException if the list is empty.

Using the list (10 marks): In the end, the main function of your MyLinkedList.cpp class should fill the linked list with 50 randomly generated Integer objects whose values are in the range from 1 to 100 and then print the list elements out. It should then be able to fill the linked list with strings " Customer 1", "Customer 2",…. Till "customer 100" and then print it again.

So, to sum up, then, you need to do the following in part 2:

1. Create the linked list interface provided.
2. Create a class, MyLinkedList which implements the Linked List interface for the SLinkedList class.
3. Fill the linked list with 50 randomly generated Integer objects whose values are in the range 1 to 100 to see if it prints correctly.
4. Fill the linked list with strings " Customer 1", "Customer 2",…. Till "customer 100" and then print the linked list again
5. Submit all of your source files, including the source code for the MyLinkedList class (MyLinkedList.cpp) and also the Linked List interface (SLinkedList.h).

Evaluation Rubric

Learning objective	Level1	Level 2	Level 3	Max Mark
Develop Algorithms (Part 1)	0-8 0-40% of the steps and syntax of the two algorithms are correct	8-16 40-80 % of the steps and syntax of the two algorithms are correct	16-20 At least 80% of the steps and syntax of the two algorithms are correct	20
Estimate efficiency (Part 1)	0-8 Number of operations is correct for 0-40% of the algorithms' steps	8-16 Number of operations is correct for 40-80% of the algorithms' steps. At least one Big-ohs is correct.	16-20 Number of operations is correct for 40-80% of the algorithms' steps. The two big-ohs are correct and a meaningful comparison is included	20
Solve a summation (Part 1)	0-5 Identify and prove the base case	5-10 Identify base case and state the hypothesis for a sub problem	10-20 Identify base case, state the hypothesis and use the hypothesis to prove it	20
Implement a data structure (Part 2)	0-8 In-class code+ a correct new function+ handling Exceptions	8-24 In-class code+ 2-3 correct new functions+ + handling Exceptions	24-30 In class code + correct four new functions++ handling Exceptions	30
Using a data structure (Part 2)	0-3 Create a single list with string or integer elements	3-8 Create an integer and string list printing one of them	8-10 Create and print two lists: a string list and an integer one	10