Lab 5
Review

Quang D. C.
dungcamquang@tdtu.edu.vn

October 25, 2021

Note

In this tutorial, we will review Linked List, Stack, Queue, Recursion and Sorting to prepare for the midterm examination.

In this Lab, lecturer will:

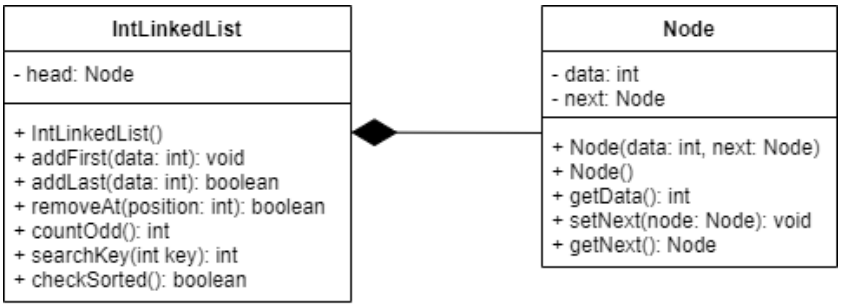
- Summarize the theory related to Linked List, Stack, Queue, Recursion and Sorting.
- Review the important knowledge for the students.

Responsibility of the students in this Lab:

- Complete all the exercises.
- Ask your lecturer if you have any question.
- Submit your solutions according to your lecturer requirement.
- Well prepare for the midterm examination.

1. Linked List

Implementing the integer Linked List following the class diagram:



(a) Method **public void addFirst(int data)**: add the new node contains *data* to the head of the linked list.

(b) Method **public boolean addLast(int data)**: this method first checks if the entered *data* already existed in the the linked list then return *false*. Otherwise, this method will add the new node as the last element of the linked list and return *true* if the node is added successfully.

(c) Method **public boolean removeAt(int position)**: if the *position* value is larger than the number of nodes in the linked list, this method will return *false*, if not, this method will remove the element at the position given by the paramater and return *true* if the node is removed. *(position 1 is the head of linked list)*

(d) Method **public int countOdd()**: return how many odd numbers there are in the linked list.

(e) Method **public int searchKey(int key)**: return the **position** of the node which contains the *key* value. If there are no elements with the *key* value, the method will return **-1**. *(position 1 is the head of linked list)*

(f) Method **public boolean checkSorted()**: return *true* if the linked list is sorted in ascending or descending order, if not, return *false*.

2. Stack and Queue

Exercise 1

Reimplemeting the **Stack** in **Lab 2**. We will use Stack<String> or Stack<Character> to solve this exercise. Giving an infix:

((9 - 2) * 6 + 7) / 7

We can transform to postfix:

9 2 - 6 * 7 + 7 /

Implement a function using **Stack** to calculate the result from postfix. This is the pseudocode:

```
   CalculatePostfix(String s)
 1 Split s into the array split_ch
 2 for ch : split_ch do
 3     if ch is an operator then
 4         a ←pop first element from stack
 5         b ←pop second element from the stack
 6         res ←b "operator" a
 7         push res into the stack
 8     ch is an operand add ch into the stack
 9 return element of stack top
```

Exercise 2

Using a **Stack** and a **Queue** to check a positive integer number n is palindrome or not. (*Examples of the palindrome number: 101, 256652, 1221, 121*)

3. Recursion

Exercise 1

(a) Implement function **public double prod_recur(int a, int b)** to calculate product of 2 numbers using recursion.

(b) Implement function **public int bin2dec(int n, int exp)** to convert a binary number (in decimal number form) to decimal number using recursion. Ex: Given n = 1000, bin2dec(n, 0) = 8

(c) Implement function **public int maxDigit(int n)** to find the largest digit in a positive integer **n** using recursion.

(d) Implement function **public int maxElement(int a[], int n)** to find the largest element in an array **a** using recursion.

(e) Implement function **public int search(int a[], int n, int key)** to find the position of the *key* in an array **a**, if *key* is not in the array, return -1, using recursion.

Exercise 2

Solve this exercise in 2 ways, using recursion and using iteration

(a) $\sum_{i=1}^{n}(2^i)$

(b) $\sum_{x=0}^{n}(\frac{x+1}{2})$

(c) $\sum_{i=1}^{n}(\frac{i!}{(i-1)!})$

(d) $\sum_{x=1}^{n}(x*(x-1))$

(e) $\prod_{x=1}^{n}(x)$

Exercise 3

For each sub-exercise below, define **2 functions**, **one uses recursion** to solve and **the other uses iteration** to solve:

(a)

$$A(n) = \begin{cases} 2, & n = 0 \\ 2 - \frac{1}{2}A(n-1), & n > 0 \end{cases}$$

(b)

$$A(n) = \begin{cases} 1, & n < 10 \\ 1 + A(n/10), & n \geq 10 \end{cases}$$

(c)

$$A(n,k) = \begin{cases} \text{n}, & k = 1 \\ n + A(n, k-1), & k > 1 \end{cases}$$

(d)

$$F(n) = \begin{cases} 0, & \text{n} = 0 \\ 1, & \text{n} = 1 \\ F(n-1) + F(n-2), & \text{otherwise} \end{cases}$$

4. Sorting

Implement **Selection Sort**, **Bubble Sort** and **Insertion Sort** again, but you need to **print to the screen the state of the array before each outer loop *for* statement completed.**

Example: We sort an array **a** in ascending order and follow this rules:

1. Selection Sort: choose minimum element
2. Bubble Sort: "bubbling up" the largest element to the right partition of the array
3. Insertion Sort: insert the number to the left partition of the array

We will have result print on screen:

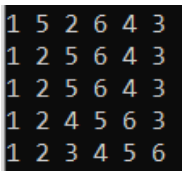
- Selection Sort - With array $a = \{3, 1, 4, 6, 2, 5\}$

```
1 3 4 6 2 5
1 2 4 6 3 5
1 2 3 6 4 5
1 2 3 4 6 5
1 2 3 4 5 6
```

- Bubble Sort - With array $a = \{5, 3, 4, 2, 6, 1\}$

```
3 4 2 5 1 6
3 2 4 1 5 6
2 3 1 4 5 6
2 1 3 4 5 6
1 2 3 4 5 6
```

- Insertion Sort - With array $a = \{5, 1, 2, 6, 4, 3\}$



THE END