BÁCH KHOA E-LEARNING

Dashboard / Courses / Học kỳ I năm học 2021-2022 (Semester 1 - Academic year 2021-2022) / Đại Học Chính Qui (Bacherlor program (Full-time study)) / Khoa Khoa học và Kỹ thuật Máy tính (Faculty of Computer Science and Engineering) / Khoa Học Máy Tính / Cấu trúc dữ liệu và giải thuật (thực hành) (CO2004)_Trần Khánh Tùng (DH_HK211) / Lab 2: Doubly Linked List + Stack + Queue + Sorting / Lab 2: Preparation

Started on	Saturday, 25 September 2021, 4:10 PM
State	Finished
Completed on	Sunday, 3 October 2021, 11:28 AM
Time taken	7 days 19 hours
Marks	4.70/6.00
Grade	**7.83** out of 10.00 (**78**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Implement methods **add, size** in template class **DLinkedList (which implements List ADT)** representing the doubly linked list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class DLinkedList {
public:
    class Node; // Forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList();
    ~DLinkedList();
    void    add(const T &e);
    void    add(int index, const T &e);
    int     size();
public:
    class Node
    {
    private:
        T data;
        Node *next;
        Node *previous;
        friend class DLinkedList<T>;

    public:
        Node()
        {
            this->previous = NULL;
            this->next = NULL;
        }

        Node(const T &data)
        {
            this->data = data;
            this->previous = NULL;
            this->next = NULL;
        }
    };

};
```

In this exercise, we have include <iostream>, <string>, <sstream> and using namespace std.

For example:

Test	**Result**

`DLinkedList<int> list;`
`int size = 10;`
`for(int idx=0; idx < size; idx++){`
`   list.add(idx);`
`}`
`cout << list.toString();`
	`[0,1,2,3,4,5,6,7,8,9]`

Test	Result

`DLinkedList<int> list;`
`int size = 10;`
`for(int idx=0; idx < size; idx++){`
`   list.add(0, idx);`
`}`
`cout << list.toString();`
	`[9,8,7,6,5,4,3,2,1,0]`

Answer: (penalty regime: 0, 0, 0, 5, 10 %)

Reset answer

```
template <class T>
void DLinkedList<T>::add(const T& e) {
    /* Insert an element into the end of the list. */
    if (count == 0)
    {
        Node* newNode = new Node(e);
        head = newNode;
        tail = newNode;
        tail->next = NULL;
        ++(this->count);
        return;
    }
    Node* newNode = new Node(e);
    tail->next = newNode;
    newNode->previous = tail;
    newNode->next = NULL;
    tail = newNode;
    ++(this->count);
    return;
}

template<class T>
void DLinkedList<T>::add(int index, const T& e) {
    /* Insert an element into the list at given index. */
    if (count == 0) {add(e);return;}
    if (index == 0)
    {
        Node* newNode = new Node(e);
        newNode->next = head;
        head->previous = newNode;
        head = newNode;
        ++(this->count);
        return;
    }
    if (index == this->count) {add(e); return;}
    int idx = 0;
    Node* front = head;
    Node* back = NULL;
    for (;front != NULL; back = front, front = front->next, ++idx)
    {
        if (idx == index)
        {
            Node* newNode = new Node (e);
            ++(this->count);
            back->next = newNode;
            newNode->next = front;
            front->previous = newNode;
            return;
        }
    }

}
template<class T>
```

	Test	Expected	Got	

✓	`DLinkedList<int> list;`
	`int size = 10;`
	`for(int idx=0; idx < size; idx++){`
	`   list.add(idx);`
	`}`
	`cout << list.toString();`
		`[0,1,2,3,4,5,6,7,8,9]`	`[0,1,2,3,4,5,6,7,8,9]`	✓

✓	`DLinkedList<int> list;`
	`int size = 10;`
	`for(int idx=0; idx < size; idx++){`
	`   list.add(0, idx);`
	`}`
	`cout << list.toString();`
		`[9,8,7,6,5,4,3,2,1,0]`	`[9,8,7,6,5,4,3,2,1,0]`	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Implement methods **get, set, empty, indexOf, contains** in template class D**LinkedList (which implements List ADT)** representing the singly linked list with type T with the initialized frame. The description of each method is given in the code.

```
template <class T>
class DLinkedList {
public:
    class Node; // Forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList();
    ~DLinkedList();
    void    add(const T &e);
    void    add(int index, const T &e);
    int     size();
    bool    empty();
    T       get(int index);
    void    set(int index, const T &e);
    int     indexOf(const T &item);
    bool    contains(const T &item);
public:
    class Node
    {
    private:
        T data;
        Node *next;
        Node *previous;
        friend class DLinkedList<T>;

    public:
        Node()
        {
            this->previous = NULL;
            this->next = NULL;
        }

        Node(const T &data)
        {
            this->data = data;
            this->previous = NULL;
            this->next = NULL;
        }
    };

};
```

In this exercise, we have include <iostream>, <string>, <sstream> and using namespace std.

For example:

Test	Result

Test	Result

`DLinkedList<int> list;`
`int size = 10;`
`for(int idx=0; idx < size; idx++){`
`  list.add(idx);`
`}`
`for(int idx=0; idx < size; idx++){`
`  cout << list.get(idx) << " |";`
`}`
	`0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |`

`DLinkedList<int> list;`
`int size = 10;`
`int value[] = {2,5,6,3,67,332,43,1,0,9};`
`for(int idx=0; idx < size; idx++){`
`  list.add(idx);`
`}`
`for(int idx=0; idx < size; idx++){`
`  list.set(idx, value[idx]);`
`}`
`cout << list.toString();`
	`[2,5,6,3,67,332,43,1,0,9]`

Answer: (penalty regime: 0, 0, 0, 5, 10 %)

Reset answer

```
template<class T>
T DLinkedList<T>::get(int index) {
    if (count == 0) return -1;
    if (index == this->count - 1) return tail->data;
    if (index == 0) return head->data;
    int idx = 0;
    for (Node* h = head; h != NULL; h = h->next, ++idx)
    {
        if (idx == index) return h->data;
    }
    return -1;
    /* Give the data of the element at given index in the list. */
}
template <class T>
void DLinkedList<T>::set(int index, const T& e) {
    if (count == 0) return;
    if (index == 0)
    {
        head->data = e;
        return;
    }
    if (index == this->count - 1)
    {
        tail->data = e;
        return;
    }
    int idx = 0;
    for (Node* h = head; h != NULL; h = h->next, ++idx)
    {
        if (idx == index)
        {
            h->data = e;
            return;
        }
    }
    /* Assign new value for element at given index in the list */
}

template<class T>
bool DLinkedList<T>::empty() {
    /* Check if the list is empty or not. */
    if (count ==0) return true;
    return false;
    return false;
}

template<class T>
int DLinkedList<T>::indexOf(const T& item) {
    int idx  = 0;
    if (count == 0) return -1;
    /* Return the first index wheter item appears in list, otherwise ret
    for (Node* h = head; h != NULL; h = h->next, ++idx)
    {
```

	Test	Expected	Got	

✓	`DLinkedList<int> list;`
	`int size = 10;`
	`for(int idx=0; idx < size; idx++){`
	`  list.add(idx);`
	`}`
	`for(int idx=0; idx < size; idx++){`
	`  cout << list.get(idx) << " |";`
	`}`
		`0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |`	`0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |`	✓

✓	`DLinkedList<int> list;`
	`int size = 10;`
	`int value[] = {2,5,6,3,67,332,43,1,0,9};`
	`for(int idx=0; idx < size; idx++){`
	`  list.add(idx);`
	`}`
	`for(int idx=0; idx < size; idx++){`
	`  list.set(idx, value[idx]);`
	`}`
	`cout << list.toString();`
		`[2,5,6,3,67,332,43,1,0,9]`	`[2,5,6,3,67,332,43,1,0,9]`	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Implement all methods in class **Stack** with template type **T**. The description of each method is written as comment in frame code.

```
#ifndef STACK_H
#define STACK_H
#include "DLinkedList.h"
template<class T>
class Stack {
protected:
    DLinkedList<T> list;
public:
    Stack() {}
    void push(T item) ;
    T pop() ;
    T top() ;
    bool empty() ;
    int size() ;
    void clear() ;
};

#endif
```

You can use all methods in class **DLinkedList** without implementing them again. The description of class **DLinkedList** is written as comment in frame code.

```
template <class T>
class DLinkedList
{
public:
    class Node;     //forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList() ;
    ~DLinkedList();
    void add(const T& e);
    void add(int index, const T& e);
    T removeAt(int index);
    bool removeItem(const T& removeItem);
    bool empty();
    int size();
    void clear();
    T get(int index);
    void set(int index, const T& e);
    int indexOf(const T& item);
    bool contains(const T& item);
};
```

For example:

Test	Result

`Stack<int> stack;`
`cout << stack.empty() << " " << stack.size();`
	1 0

`Stack<int> stack;`
 	
`int item[] = { 3, 1, 4, 5, 2, 8, 10, 12 };`
`for (int idx = 0; idx < 8; idx++) stack.push(item[idx]);`
 	
`assert(stack.top() == 12);`
 	
`stack.pop();`
`stack.pop();`
 	
`cout << stack.top();`
	8

Answer: (penalty regime: 0, 0, 5, 10 %)

Reset answer

```

void  push(T item) {
    // TODO: Push new element into the top of the stack
    list.add(0,item);
}

T  pop() {
    // TODO: Remove an element on top of the stack
     T temp =    list.removeAt(0);
     return temp;
}

T  top() {
    // TODO: Get value of the element on top of the stack
        return list.get(0);
}

bool  empty() {
    // TODO: Determine if the stack is empty
        if (list.size() == 0) return true;
        return false;
}

int  size() {
    // TODO: Get the size of the stack
    return list.size();
}

void  clear() {
    // TODO: Clear all elements of the stack
    list.clear();
}
```

	Test	Expected	Got	

✓	`Stack<int> stack;`
	`cout << stack.empty() << " " << stack.size();`
		1 0	1 0	✓

✓	`Stack<int> stack;`
	
	`int item[] = { 3, 1, 4, 5, 2, 8, 10, 12 };`
	`for (int idx = 0; idx < 8; idx++) stack.push(item[idx]);`
	
	`assert(stack.top() == 12);`
	
	`stack.pop();`
	`stack.pop();`
	
	`cout << stack.top();`
		8	8	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Implement all methods in class **Queue** with template type **T**. The description of each method is written as comment in frame code.

```
#ifndef QUEUE_H
#define QUEUE_H
#include "DLinkedList.h"
template<class T>
class Queue {
protected:
    DLinkedList<T> list;
public:
    Queue() {}
    void push(T item) ;
    T pop() ;
    T top() ;
    bool empty() ;
    int size() ;
    void clear() ;
};

#endif /* QUEUE_H */
```

You can use all methods in class **DLinkedList** without implementing them again. The description of class **DLinkedList** is written as comment in frame code.

```
template <class T>
class DLinkedList
{
public:
    class Node;     //forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList() ;
    ~DLinkedList();
    void add(const T& e);
    void add(int index, const T& e);
    T removeAt(int index);
    bool removeItem(const T& removeItem);
    bool empty();
    int size();
    void clear();
    T get(int index);
    void set(int index, const T& e);
    int indexOf(const T& item);
    bool contains(const T& item);
};
```

For example:

Test	Result

Test	Result

`Queue<int> queue;`
`    assert(queue.empty());`
`    assert(queue.size() == 0);`	

Answer: (penalty regime: 0, 0, 5, 10 %)

Reset answer

```
void push(T item) {
    // TODO: Push new element into the end of the queue
    list.add(item);
}

T pop() {
    // TODO: Remove an element in the head of the queue
    T temp = list.removeAt(0);
    return temp;
}

T top() {
    // TODO: Get value of the element in the head of the queue
    return list.get(0);
}

bool empty() {
    // TODO: Determine if the queue is empty
    if (list.size() == 0 ) return true;
    return false;

}

int size() {
    // TODO: Get the size of the queue
        return list.size();
}

void clear() {
    // TODO: Clear all elements of the queue
    list.clear();
}
```

	Test

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Incorrect

Mark 0.00 out of 1.00

Implement static methods **Partition** and **QuickSort** in class Sorting to sort an array in ascending order.

```
#ifndef SORTING_H
#define SORTING_H
#include <sstream>
#include <iostream>
#include <type_traits>
using namespace std;
template <class T>
class Sorting {
private:
    static T* Partition(T* start, T* end) ;
public:
    static void QuickSort(T* start, T* end) ;
};
#endif /* SORTING_H */
```

You can read the pseudocode of the algorithm used to in method Partition in the below image.

```
ALGORITHM  HoarePartition(A[l..r])
    //Partitions a subarray by Hoare's algorithm, using the first element
    //          as a pivot
    //Input: Subarray of array A[0..n − 1], defined by its left and right
    //          indices l and r (l < r)
    //Output: Partition of A[l..r], with the split position returned as
    //          this function's value
    p ← A[l]
    i ← l; j ← r + 1
    repeat
        repeat i ← i + 1 until A[i] ≥ p
        repeat j ← j − 1 until A[j] ≤ p
        swap(A[i], A[j])
    until i ≥ j
    swap(A[i], A[j])   //undo last swap when i ≥ j
    swap(A[l], A[j])
    return j
```

For example:

Test	Result

`int array[] = { 3, 5, 7, 10 ,12, 14, 15, 13, 1, 2, 9, 6, 4, 8, 11, 16, 17, 18, 20, 19 };`
`cout << "Index of pivots: ";`
`Sorting<int>::QuickSort(&array[0], &array[20]);`
`cout << "\n";`
`cout << "Array after sorting: ";`
`for (int i : array) cout << i << " ";`
	`Index of pivots: 2 0 0 6 1 0 2 1 0 0 2 1 0 0 0 0 0 0 1 0`
	`Array after sorting: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20`

Answer: (penalty regime: 0, 0, 5, 10, 15, ... %)

Reset answer

```

static T* Partition(T* start, T* end) {
    // TODO: return the pointer which points to the pivot after rearrange t
    T pivot = *end; // pivot
    //T i = (low - 1); // Index of smaller element and indicates the right

    while (start != NULL)
    while (start != NULL)
    {
        // If current element is smaller than the pivot
        if (*start < pivot)
        {
            start = start + 1; // increment index of smaller element
            swap(*start, *end);
        }
    }
    swap(*(start+1), *end);
    return (start+1);
}
static void QuickSort(T* start, T* end) {
    if (*start < *end)
    {
        /* pi is partitioning index, arr[p] is now
        at right place */
        T* pi = Partition(start, end);

        // Separately sort elements before
        // partition and after partition
        QuickSort(start, pi - 1);
        QuickSort(pi + 1, end);
    }
    // TODO
    // In this question, you must print out the index of pivot in subarray

}
```

while (start != NULL)

	Test	Expected	Got	

Testing was aborted due to error.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

Question **6**

Partially correct

Mark 0.70 out of 1.00

Implement method bubbleSort() in class SLinkedList to sort this list in ascending order. After each bubble, we will print out a list to check (using printList).

```
#include <iostream>
#include <sstream>
using namespace std;

template <class T>
class SLinkedList {
public:
    class Node; // Forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    SLinkedList()
    {
      this->head = nullptr;
      this->tail = nullptr;
      this->count = 0;
    }
    ~SLinkedList(){};
    void add(T e)
    {
        Node *pNew = new Node(e);

        if (this->count == 0)
        {
            this->head = this->tail = pNew;
        }
        else
        {
            this->tail->next = pNew;
            this->tail = pNew;
        }

        this->count++;
    }
    int size()
    {
        return this->count;
    }
    void printList()
    {
        stringstream ss;
        ss << "[";
        Node *ptr = head;
        while (ptr != tail)
        {
            ss << ptr->data << ",";
            ptr = ptr->next;
        }

        if (count > 0)
            ss << ptr->data << "]";
        else
            ss << "]";
        cout << ss.str() << endl;
    }
public:
    class Node {
    private:
        T data;
        Node* next;
        friend class SLinkedList<T>;
    public:
        Node() {
            next = 0;
        }
        Node(T data) {
            this->data = data;
            this->next = nullptr;
        }
    };

    void bubbleSort();
};
```

For example:

Test	Result

`int arr[] = {9, 2, 8, 4, 1};`
`SLinkedList<int> list;`
`for(int i = 0; i <int(sizeof(arr))/4;i++)`
`    list.add(arr[i]);`
`list.bubbleSort();`
	`[2,8,4,1,9]`
	`[2,4,1,8,9]`
	`[2,1,4,8,9]`
	`[1,2,4,8,9]`

Answer: (penalty regime: 0, 0, 5, 10, 15, ... %)

Reset answer

```
template <class T>
void SLinkedList<T>::bubbleSort()
{
     if (count <= 1) {printList(); return;}
    bool cmp = true;
    for (int i = 0; i < count; ++i)
    {
         cmp = false;
        Node* curr = head->next;
        Node* prev = head;
        for (; curr!= NULL; prev = curr, curr = curr->next)
        {
            if (prev->data > curr->data)
            {
                T temp = prev->data;
                prev->data = curr->data;
                curr->data = temp;
                cmp = true;
            }
        }
         if (cmp == false && i !=  0) break;
         else printList();
    }
}
```

	Test	Expected	Got	
✓	int arr[] = {9, 2, 8, 4, 1};	[2 8 4 1 9]	[2 8 4 1 9]	✓

Partially correct

Marks for this submission: 0.60/1.00. Accounting for previous tries, this gives **0.70/1.00**.

◄ Lab 1: Singly Linked List

Jump to...

Lab 2: Doubly LL, Stack and Queue ►