Creative Software Programming, Assignment 12-2

Handed out: Dec 2, 2021

Due: 23:59, Dec 7, 2021 **(NO SCORE for late submissions!)**

- *Only files submitted by **git push to this course project at** https://hconnect.hanyang.ac.kr (<Year>_<Course no.>_<Class code>/<Year>_<Course no.>_<Student ID>.git) will be scored.*
- *Place your files under the directory structure **<Assignment name>/<Problem no.>/<your files>** just like the following example.*
  - ***DO NOT push CMake intermediate output files** (CMakeCache.txt, cmake_install.cmake, Makefile, CMakeFiles/*).*
  - ***DO NOT use cmake_minimum_required()** command in CMakeLists.txt.*

```
+ 2020_ITE0000_2019000001
  + 2-1/
    + 1/
      - 1.cpp
      - CMakeLists.txt
    + 2/
      - 2.cpp
      - CMakeLists.txt
    + …
```

- *The submission time is determined not when the commit is made **but when the git push is made**.*
- *Your files must be committed to the **master branch**. Otherwise, it will not be scored.*
- *Your program should output correct results even for inputs other than those used in the example.*
- *Basically, assignments are scored based on the output results. If it is not possible to check whether a requirement is implemented because the output is not correct, no score is given for the requirement, even if it is implemented internally. However, even if the output result is correct, no score is given for a requirement if the internal implementation does not satisfy the requirement.*

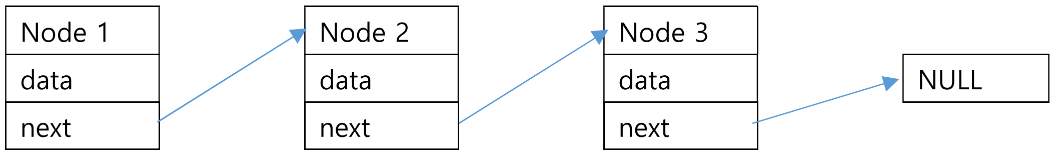
1. Write a program for templated singly linked list.

   A. The list stores the following Node class template instances.

```
template <class T>
class Node
{
        public:
                T data;
                Node<T>* next;
}
```

   B.

   C. Diagram of singly linked list:



   D. Complete the following class template List that implements singly linked list.

      i. Hint: Set the next of the last node of the liked list to NULL.

```
template <class T>
class List {
private:
    Node<T> *head;
public:
    List() : head(NULL) {};
    ~List();//free resources
    List(T* arr, int n_nodes);//create a list with n_nodes
    void insert_at(int idx, const T& data);
    void remove_at(int idx);
    void pop_back();
    void push_back(const T& val);
    void pop_front();
    void push_front(const T& val);
    friend ostream& operator<<(ostream& out, List& rhs);//print out nodes
};
```

   E.

   F. Use the following main function to test your List class template.

G.
```
int main(){
            int array[5] = {12, 7, 9, 21, 13 };
            List<int> li(array, 5);
            cout<< li << endl; //12,7,9,21,13

            li.pop_back();
            cout<< li << endl; //12,7,9,21

            li.push_back(15);
            cout<< li << endl; //12,7,9,21,15

            li.pop_front();
            cout<< li << endl; //7,9,21,15

            li.push_front(8);
            cout<< li << endl; //8,7,9,21,15

            li.insert_at(4, 19);
            cout<< li << endl; //8,7,9,21,19,15

            li.remove_at(1);
            cout<< li << endl; //8,9,21,19,15

            return 0;
      }
```

H. Input: None

I. Output: Printed results of running main() function

J. Files to submit:

   i. main.cpp – Use the given main() code as is.

   ii. list.h – Implementation of class template List (Write the function bodies in the header file)

   iii. A CMakeLists.txt to generate the executable

```
$ ./list
12,7,9,21,13
12,7,9,21
12,7,9,21,15
7,9,21,15
8,7,9,21,15
8,7,9,21,19,15
8,9,21,19,15
$
```

2. Write a program that utilizes the previous problem.
   A. Merge two lists in order.
   B. Create two List objects with given arrays.
   C. Note that:
      i. Input arrays are already sorted.
   D. Use the following main function to test your program.

```
int main(){
                int array[5] = {1,2,4,5,6 };
                int array2[4] = {1,3,5,7 };

                List<int> li(array, 5);
                List<int> li2(array2, 4);

                //implement here

                cout << li << endl; // 1,1,2,3,4,5,5,6,7

                return 0;
        }
```

   E. Input: None
   F. Output: Printed results of running main() function
   G. Files to submit:
      i. main.cpp – Use the given main() code as is.
      ii. list.h – Implementation of class template List (Write the function bodies in the header file)
      iii. A CMakeLists.txt to generate the executable

```
$ ./list
1,1,2,3,4,5,5,6,7
$
```