CSED 232 Object-Oriented Programming (Spring 2024)

Programming Assignment #3

Due date: 4 월 5 일 23 시 59 분 59 초

담당 조교: 김유지 (ugkim@postech.ac.kr)

[안내사항]

1. 모든 문제는 C++의 standard 입출력(i.e. std::cin, std::cout)을 기본으로 합니다.
2. 프로그램 기능 점수 기준은 채점용 testcase 통과 여부입니다.
    1) 채점용 testcase 들의 점수는 동일하지 않습니다. 문제에서 요구하는 기능별 난이도에 따라 배점이 다릅니다.
    2) 채점용 testcase 는 비공개입니다.
3. 미리 정의되어 있는 C/C++ library functions and classes 는 사용할 수 없습니다. 예외적으로 limits 헤더파일 include 는 가능합니다.

[감점]

1. 제출 기한이 지나면 얻은 총점의 20% 감점
2. 하루(24 시간) 늦을 때마다 추가 20%씩 감점
    a. 1 일 이내: 20% 감점, 2 일 이내: 40% 감점, 3 일 이내: 60% 감점, 4 일 이내 80% 감점
    b. 4 일 이상: 0 점
3. 컴파일이 정상적으로 되지 않을 경우 프로그램 기능 점수 0 점

[제출방식]

채점 환경은 Windows Visual Studio 2022 입니다. 파일을 업로드하실 때, **개발환경 파일의 "파일 제출"페이지에 써 있는 대로** 맞춰 올려 주시기 바랍니다. 폴더명은 문제#_학번(e.g. prob1_20230000)으로 만들어 주시기 바랍니다. 또한 문제 폴더 안에 각 문제에 해당하는 Report(e.g. prob1_20230000_report)도 첨부하여 zip 파일로 압축한 후 제출해 주시기 바랍니다. 제출 포맷이 맞지 않는 경우 감점이 있습니다.

이번 과제에서는 소스(.cpp) 파일과 헤더(.h) 파일을 분리하여 작성합니다. 아래 문제에 제시된 구조체와 함수는 헤더 파일에, 실제 구현은 cpp 파일에 구분해서 진행해주시면 됩니다. 입출력 예시에 나타나있는 것처럼 채점은 헤더파일을 import 하여 진행됩니다.

제출 시 파일명은 문제.cpp, 문제.h 로 제출 부탁드립니다(e.g., prob1.cpp, prob1.h). header 와 cpp 파일 모두 제출 부탁드리고, 테스트 시 활용한 다른 코드는 삭제 부탁드립니다.

문제마다 따로 프로젝트를 생성하고, 따로 압축하여 제출해 주시기 바랍니다. 즉, **총 2 개의 파일을 제출**하셔야 합니다. 제출은 반드시 PLMS 를 통해 제출해주시기 바랍니다. 이메일 제출은 받지 않습니다. 제출 기한을 기준으로 4 일(4 월 9 일 23 시 59 분 59 초)이 경과한 이후에는 0 점이므로 PLMS 에서도 과제 제출 받지 않습니다.

제출파일 예시) prob1_20230000.zip, prob2_20230000.zip

[채점기준]

1. **프로그램 기능 – 50%**
    1) 프로그램이 요구 사항을 모두 만족하면서 올바로 실행되는가?
2. **프로그램 설계 및 구현 – 35%**
    1) 요구 사항을 만족하기 위한 프로그램(변수, 함수, 알고리즘 등) 설계가 적절한가?
    2) 각 문제에서 제시한 세부 조건의 유의사항을 모두 만족하였는가?
    3) 입력과 출력이 주어진 형식에 맞게 프로그램이 잘 작동하는가?
3. **프로그램 가독성 – 5%**
    1) 프로그램이 읽기 쉽고 이해하기 쉽게 작성되었는가?
    2) 변수 및 함수 명이 무엇을 의미하는지 파악하기 쉬운가?
    3) 프로그램의 소스 코드를 이해하기 쉽도록 주석을 잘 작성하였는가?
4. **보고서 구성 및 내용, 양식 – 10%**
    1) 보고서는 적절한 내용으로 이해하기 쉽고 보기 좋게 잘 작성되었는가?
    2) 보고서의 양식을 잘 따랐는가?
    3) 각 문제에서 제시한 질문이 있다면, 그에 대한 답변이 충분하고 적절한가?

[주의사항]

다른 사람의 프로그램이나 인터넷 등에 있는 프로그램을 단순히 복사(copy)하거나 수정해서 제출하면 부정행위로 간주됩니다. 부정행위 적발 시 'F' 학점을 받을 수 있으며, 학과에서 정한 기준에 따라 추가적인 불이익이 있을 수 있습니다.

주석을 작성할 시 영어로 작성해주시기 바랍니다. (한글주석 작성 시 다른 OS 에서 컴파일이 안되는 상황이 발생합니다

문제 1 번: OrderedList 구현 (배점 70 점)

[문제 설명]

아래 설명을 읽고, 간단한 list 구현에서 발전되어 element 들이 ordered 되어있는 **OrderedList** 를 구현한다. OrderedList 는 struct 를 사용하여 구현한다. OrderedList 는 integer value 에 대해서만 동작한다.

[프로그램 기능]

- OrderedList 의 주요 기능으로는 element 를 추가하고, 삭제하고, element 에 접근하고, value 가 list 에 존재하는지 확인하는 것이다.
- 중요한 기능으로는 새로운 element 를 추가할 때, increasing order 에 맞는 적절한 위치에 추가해야 한다.
- 예를 들면, "5", "7", "6"의 순서로 list 에 element 를 추가했다면, list elements 들은 "5", "6", "7"의 순서로 저장되어야 한다.

[세부조건]

1) OrderedList 를 나타내는 struct 를 구현한다. OrderedList 는 다음과 같은 멤버변수를 가진다.

- int m_size: OrderedList instance 사이즈
- Node* head: OrderedList instance 의 시작 node (linked list 형태의 struct Node 구현 필요)

2) OrderedList 에 element 를 추가하는 함수

- void add(OrderedList* ordered, int v);
- void add(OrderedList* ordered, const int* arr, int size);

첫 번째 function 은 new element v 를 increasing order 를 유지하면서 list 에 추가한다.

두 번째 function 은 integer values 를 가지는 array 를 받아서 올바른 order 를 유지하면서 list 에 값들을 추가한다.

3) OrderedList 에 element 를 삭제하는 함수

- void remove(OrderedList* ordered, int index)

해당 function 은 주어진 idx 에 해당하는 element 를 삭제한다

4) 생성된 list instance 의 element 수를 반환하는 함수

- int size(OrderedList* ordered);

add/remove 함수 내부에서 m_size 를 적절하게 변경해주는 형태로 구현한다.

5) 주어진 element 가 존재하는지 확인하는 함수

- bool contains(OrderedList* ordered, int v);

예를 들면, "7", 6", "5"를 순서대로 "OrderedList test"에 더해주었을 때, "contains(&test, 5)"는 true 를 반환한다. "contains(&test, 10)"의 경우 false 를 반환한다.

6) 주어진 index 에 element 가 존재하는지 확인하는 함수

- int getValue(OrderedList* ordered, int idx);

생성된 OrderedList instance 에서 주어진 idx 에 위치하는 value 를 반환한다.

Out-of-index 의 경우에는 minimum integer(i.e., std::numeric_limits<*int*>::min())을 반환한다(이를 위해서는 limits 헤더 파일 include 필요).

[입출력 예시]

prob1_main.cpp

```
#include "prob1.h"
```

```
#include <iostream>

void simpleTest1(OrderedList* o) {
    add(o, 5);
    add(o, 4);
    add(o, 3);
    add(o, 80);
    add(o, 50);

    for (int i = 0; i < o->m_size; ++i) {
        std::cout << getValue(o, i) << ", ";
    }

    std::cout << std::endl;
}

void simpleTest2(OrderedList* o) {
    int vals[] = { 10, 20, 35, 35, 10 };
    add(o, vals, sizeof(vals) / sizeof(int));

    for (int i = 0; i < o->m_size; ++i) {
        std::cout << getValue(o, i) << ", ";
    }

    std::cout << std::endl;
}

void simpleTest3(OrderedList* o) {
    remove(o, 3);

    for (int i = 0; i < o->m_size; ++i) {
        std::cout << getValue(o, i) << ", ";
    }

    std::cout << std::endl;
}

void simpleTest4(OrderedList* o) {
    std::cout << std::boolalpha << contains(o, 20) << std::endl;
    std::cout << std::boolalpha << contains(o, 40) << std::endl;
}

void simpleTest5(OrderedList* o) {
    std::cout << size(o) << std::endl;
}
```

```cpp
int main()
{
    OrderedList orderedList;
    std::cout << "<<Simple Test 1>>" << std::endl;
    simpleTest1(&orderedList);

    std::cout << "<<Simple Test 2>>" << std::endl;
    simpleTest2(&orderedList);

    std::cout << "<<Simple Test 3>>" << std::endl;
    simpleTest3(&orderedList);

    std::cout << "<<Simple Test 4>>" << std::endl;
    simpleTest4(&orderedList);

    std::cout << "<<Simple Test 5>>" << std::endl;
    simpleTest5(&orderedList);

    return 0;
}
```

출력 결과

```
<<Simple Test 1>>
3, 4, 5, 50, 80,
<<Simple Test 2>>
3, 4, 5, 10, 10, 20, 35, 35, 50, 80,
<<Simple Test 3>>
3, 4, 5, 10, 20, 35, 35, 50, 80,
<<Simple Test 4>>
true
false
<<Simple Test 5>>
9
```

문제 2 번: OrderedSet 구현 (배점 30 점)

[문제 설명]

아래 설명을 읽고, 문제 1 번의 **OrderedList** 에서 겹치는 element 가 없는 발전된 형태인 **OrderedSet** 을 구현한다. OrderedSet 은 struct 를 사용하여 구현한다. 문제 1 번과 겹치는 함수에 대해서는 그대로 가져오거나 수정하여 사용 가능하다. OrderedSet 는 integer value 에 대해서만 동작한다.

[프로그램 기능]

- OrderedSet 은 OrderedList 와 동일한 기능을 수행한다. 유일한 다른 점으로는, 겹치는 element 를 허용하지 않는다.
- 예를 들면, "5", "5", "7"의 순서로 list 에 element 를 추가했다면, list elements 들은 "5", "7"의 순서로 저장되어야 한다.

[세부조건]

1) OrderedSet 를 나타내는 struct 를 구현한다. OrderedSet 는 다음과 같은 멤버변수를 가진다.

- int m_size: OrderedSet instance 사이즈
- Node* head: OrderedSet instance 의 시작 node (linked list 형태의 struct Node 구현 필요)

2) OrderedSet 에 element 를 추가하는 함수

- void add(OrderedSet * ordered, int v);
- void add (OrderedSet * ordered, const int* arr, int size);

첫 번째 function 은 new element v 를 increasing order 를 유지하면서 list 에 추가한다.

두 번째 function 은 integer values 를 가지는 array 를 받아서 올바른 order 를 유지하면서 list 에 값들을 추가한다.

주의해야 할 점으로는 겹치는 element 는 추가하지 않는다. (hint: 첫 번째 function 을 활용하여 두 번째 function 을 작성할 경우, 첫 번째 function 에서만 duplication 을 신경쓰면 된다.)

3) OrderedSet 에 element 를 삭제하는 함수

- void remove(OrderedSet * ordered, int index)

해당 function 은 주어진 idx 에 해당하는 element 를 삭제한다

4) 생성된 list instance 의 element 수를 반환하는 함수

- int size(OrderedSet * ordered);

add/remove 함수 내부에서 m_size 를 적절하게 변경해주는 형태로 구현한다.

5) 주어진 element 가 존재하는지 확인하는 함수

- bool contains(OrderedSet * ordered, int v);

예를 들면, "7", 6", "5"를 순서대로 "OrderedSet test"에 더해주었을 때, "contains(&test, 5)"는 true 를 반환한다. "contains(&test, 10)"의 경우 false 를 반환한다.

주어진 element 인 v 값이 OrderedSet instance 에 겹치는지 확인하기 위해 add 함수 내부에서 적절하게 활용 가능하다.

6) 주어진 index 에 element 가 존재하는지 확인하는 함수

- int getValue(OrderedSet * ordered, int idx);

생성된 OrderedSet instance 에서 주어진 idx 에 위치하는 value 를 반환한다.

Out-of-index 의 경우에는 minimum integer(i.e., std::numeric_limits<*int*>::min())을 반환한다(이를 위해서는 limits 헤더 파일 include 필요).

[입출력 예시]

prob2_main.cpp

```cpp
#include "prob2.h"
#include <iostream>

void simpleTest1(OrderedSet* o) {
    add(o, 5);
    add(o, 5);
    add(o, 5);
    add(o, 4);
    add(o, 3);

    for (int i = 0; i < o->m_size; ++i) {
        std::cout << getValue(o, i) << ", ";
    }

    std::cout << std::endl;
}

void simpleTest2(OrderedSet* o) {
    int vals[] = { 10, 20, 35, 35, 10 };
    add(o, vals, sizeof(vals) / sizeof(int));

    for (int i = 0; i < o->m_size; ++i) {
        std::cout << getValue(o, i) << ", ";
    }

    std::cout << std::endl;
}

void simpleTest3(OrderedSet* o) {
    remove(o, 3);

    for (int i = 0; i < o->m_size; ++i) {
        std::cout << getValue(o, i) << ", ";
    }

    std::cout << std::endl;
}
```

```
void simpleTest4(OrderedSet* o) {
    std::cout << std::boolalpha << contains(o, 20) << std::endl;
    std::cout << std::boolalpha << contains(o, 40) << std::endl;
}

void simpleTest5(OrderedSet* o) {
    std::cout << size(o) << std::endl;
}

int main()
{
    OrderedSet orderedSet;
    std::cout << "<<Simple Test 1>>" << std::endl;
    simpleTest1(&orderedSet);

    std::cout << "<<Simple Test 2>>" << std::endl;
    simpleTest2(&orderedSet);

    std::cout << "<<Simple Test 3>>" << std::endl;
    simpleTest3(&orderedSet);

    std::cout << "<<Simple Test 4>>" << std::endl;
    simpleTest4(&orderedSet);

    std::cout << "<<Simple Test 5>>" << std::endl;
    simpleTest5(&orderedSet);

    return 0;
}
```

출력 결과

```
<<Simple Test 1>>
3, 4, 5,
<<Simple Test 2>>
3, 4, 5, 10, 20, 35,
<<Simple Test 3>>
3, 4, 5, 20, 35,
<<Simple Test 4>>
true
false
<<Simple Test 5>>
5
```