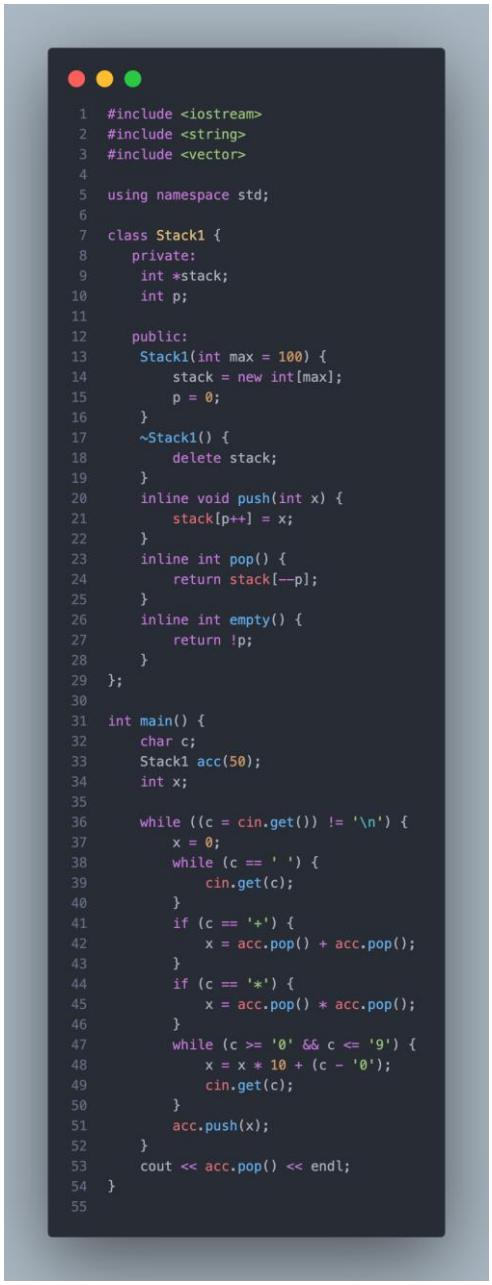
컴퓨터알고리즘과실습 2주차 실습

2020112046 이재원

2 – 1.



```cpp
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Stack1 {
   private:
    int *stack;
    int p;

   public:
    Stack1(int max = 100) {
        stack = new int[max];
        p = 0;
    }
    ~Stack1() {
        delete stack;
    }
    inline void push(int x) {
        stack[p++] = x;
    }
    inline int pop() {
        return stack[--p];
    }
    inline int empty() {
        return !p;
    }
};

int main() {
    char c;
    Stack1 acc(50);
    int x;

    while ((c = cin.get()) != '\n') {
        x = 0;
        while (c == ' ') {
            cin.get(c);
        }
        if (c == '+') {
            x = acc.pop() + acc.pop();
        }
        if (c == '*') {
            x = acc.pop() * acc.pop();
        }
        while (c >= '0' && c <= '9') {
            x = x * 10 + (c - '0');
            cin.get(c);
        }
        acc.push(x);
    }
    cout << acc.pop() << endl;
}
```

```cpp
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Stack1 {
   private:
    int *stack;
    int p;

   public:
    Stack1(int max = 100) {
        stack = new int[max];
        p = 0;
    }
    ~Stack1() {
        delete stack;
    }
    inline void push(int x) {
        stack[p++] = x;
    }
    inline int pop() {
        return stack[--p];
    }
    inline int empty() {
        return !p;
    }
};

int main() {
    char c;
    Stack1 acc(50);
    int x;

    while ((c = cin.get()) != '\n') {
        x = 0;
        while (c == ' ') {
            cin.get(c);
        }
        if (c == '+') {
            x = acc.pop() + acc.pop();
        }
        if (c == '*') {
            x = acc.pop() * acc.pop();
        }
        while (c >= '0' && c <= '9') {
            x = x * 10 + (c - '0');
            cin.get(c);
        }
        acc.push(x);
    }
    cout << acc.pop() << endl;
}
```

1번 문제는 배열을 이용해 Stack을 구현하고, 후위 표기법(postfix)으로 된 식을 계산해보는 간단한 문제였다.

push, pop 모두 O(1) 의 시간이 걸리기에, 전체 시간 복잡도는 입력된 숫자 or 연산자의 수를 n 이라 할 때 , O(n).

Space Complexity도 O(n).

2 – 2.

```cpp
#include <iostream>

using namespace std;

class Stack2 {
   private:
    struct node {
        int data;
        node *next;
        node *prev;
    };
    struct node *head, *cursor;

   public:
    Stack2(int max = 100) {
        head = NULL;
        cursor = NULL;
    }
    ~Stack2() {
        head = NULL;
        cursor = NULL;
    }
    void push(int x) {
        node *temp = new node;
        temp->data = x;
        temp->next = NULL;
        temp->prev = NULL;
        if (head == NULL) {
            head = temp;
            cursor = temp;
        } else {
            cursor->next = temp;
            temp->prev = cursor;
            cursor = temp;
        }
    }
    int pop() {
        if (head == NULL) {
            cout << "Stack is empty" << endl;
            return 0;
        } else {
            int x = cursor->data;
            if (cursor == head) {
                delete cursor;
                cursor = NULL;
                head = NULL;
            } else {
                cursor = cursor->prev;
                delete cursor->next;
                cursor->next = NULL;
            }
            return x;
        }
    }
    void empty() {
        while (head != NULL) {
            pop();
        }
    }
};

int main() {
    char c;
    Stack2 acc(50);
    int x, y;

    while ((c = cin.get()) != '\n') {
        x = 0;
        while (c == ' ') {
            cin.get(c);
        }
        if (c == '+') {
            x = acc.pop() + acc.pop();
        }
        if (c == '*') {
            x = acc.pop() * acc.pop();
        }
        if (c == '-') {
            x = acc.pop();
            y = acc.pop();
            x = y - x;
        }
        if (c == '/') {
            x = acc.pop();
            y = acc.pop();
            x = y / x;
        }
        while (c >= '0' && c <= '9') {
            x = x * 10 + (c - '0');
            cin.get(c);
        }
        acc.push(x);
    }
```

```cpp
#include <iostream>

using namespace std;

class Stack2 {
   private:
    struct node {
        int data;
        node *next;
        node *prev;
    };
    struct node *head, *cursor;

   public:
    Stack2(int max = 100) {
        head = NULL;
        cursor = NULL;
    }
    ~Stack2() {
        head = NULL;
        cursor = NULL;
    }
    void push(int x) {
        node *temp = new node;
        temp->data = x;
        temp->next = NULL;
        temp->prev = NULL;
        if (head == NULL) {
            head = temp;
            cursor = temp;
        } else {
            cursor->next = temp;
            temp->prev = cursor;
            cursor = temp;
        }
    }
    int pop() {
        if (head == NULL) {
            cout << "Stack is empty" << endl;
            return 0;
        } else {
            int x = cursor->data;
            if (cursor == head) {
                delete cursor;
                cursor = NULL;
                head = NULL;
            } else {
                cursor = cursor->prev;
                delete cursor->next;
                cursor->next = NULL;
            }
            return x;
        }
    }
```

```
    void empty() {
        while (head != NULL) {
            pop();
        }
    }
};

int main() {
    char c;
    Stack2 acc(50);
    int x, y;

    while ((c = cin.get()) != '\n') {
        x = 0;
        while (c == ' ') {
            cin.get(c);
        }
        if (c == '+') {
            x = acc.pop() + acc.pop();
        }
        if (c == '*') {
            x = acc.pop() * acc.pop();
        }
        if (c == '-') {
            x = acc.pop();
            y = acc.pop();
            x = y - x;
        }
        if (c == '/') {
            x = acc.pop();
            y = acc.pop();
            x = y / x;
        }
        while (c >= '0' && c <= '9') {
            x = x * 10 + (c - '0');
            cin.get(c);
        }
        acc.push(x);
    }
    cout << acc.pop() << endl;
}
```

2번 문제는 1번 문제와 동일하게 후위 표기법으로 쓰인 연산식을 계산하는 문제였는데,

Stack을 포인터를 이용하여 구현해야 했다.

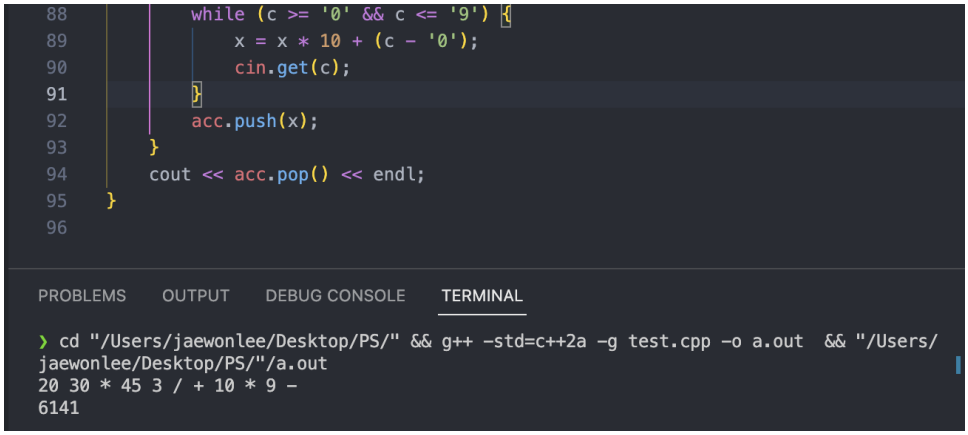
Singly Linked List로 Stack을 구현하면,

pop 과정에서 새로운 tail을 찾을 때마다 O(n)의 시간이 걸리므로,

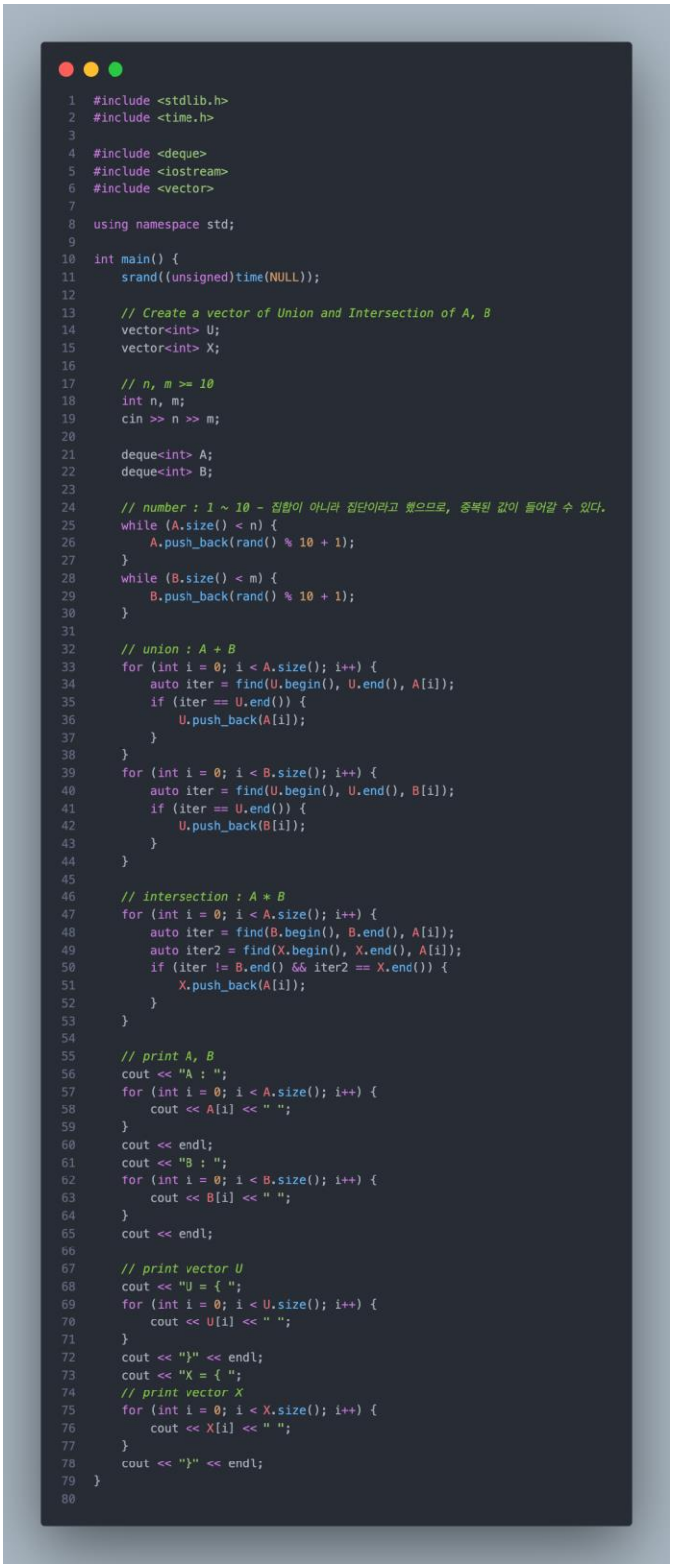
push와 pop 모두 O(1)로 끝낼 수 있는 Doubly Linked List로 Stack을 구현했다.

그리하여 1번과 동일하게 전체 시간 복잡도는 O(n).

당연히 Space Complexity도 O(n).



2 – 3.



```cpp
#include <stdlib.h>
#include <time.h>

#include <deque>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    srand((unsigned)time(NULL));

    // Create a vector of Union and Intersection of A, B
    vector<int> U;
    vector<int> X;

    // n, m >= 10
    int n, m;
    cin >> n >> m;

    deque<int> A;
    deque<int> B;

    // number : 1 ~ 10 - 집합이 아니라 집단이라고 했으므로, 중복된 값이 들어갈 수 있다.
    while (A.size() < n) {
        A.push_back(rand() % 10 + 1);
    }
    while (B.size() < m) {
        B.push_back(rand() % 10 + 1);
    }

    // union : A + B
    for (int i = 0; i < A.size(); i++) {
        auto iter = find(U.begin(), U.end(), A[i]);
        if (iter == U.end()) {
            U.push_back(A[i]);
        }
    }
    for (int i = 0; i < B.size(); i++) {
        auto iter = find(U.begin(), U.end(), B[i]);
        if (iter == U.end()) {
            U.push_back(B[i]);
        }
    }

    // intersection : A * B
    for (int i = 0; i < A.size(); i++) {
        auto iter = find(B.begin(), B.end(), A[i]);
        auto iter2 = find(X.begin(), X.end(), A[i]);
        if (iter != B.end() && iter2 == X.end()) {
            X.push_back(A[i]);
        }
    }

    // print A, B
    cout << "A : ";
    for (int i = 0; i < A.size(); i++) {
        cout << A[i] << " ";
    }
    cout << endl;
    cout << "B : ";
    for (int i = 0; i < B.size(); i++) {
        cout << B[i] << " ";
    }
    cout << endl;

    // print vector U
    cout << "U = { ";
    for (int i = 0; i < U.size(); i++) {
        cout << U[i] << " ";
    }
    cout << "}" << endl;
    cout << "X = { ";
    // print vector X
    for (int i = 0; i < X.size(); i++) {
        cout << X[i] << " ";
    }
    cout << "}" << endl;
}

```

```cpp
#include <stdlib.h>
```

```cpp
#include <time.h>

#include <deque>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    srand((unsigned)time(NULL));

    // Create a vector of Union and Intersection of A, B
    vector<int> U;
    vector<int> X;

    // n, m >= 10
    int n, m;
    cin >> n >> m;

    deque<int> A;
    deque<int> B;

    // number : 1 ~ 10000 - 집합이 아니라 집단이라고 했으므로, 중복된 값이 들어갈 수 있다.
    while (A.size() < n) {
        A.push_back(rand() % 10 + 1);
    }
    while (B.size() < m) {
        B.push_back(rand() % 10 + 1);
    }

    // union : A + B
    for (int i = 0; i < A.size(); i++) {
        auto iter = find(U.begin(), U.end(), A[i]);
        if (iter == U.end()) {
            U.push_back(A[i]);
        }
    }
    for (int i = 0; i < B.size(); i++) {
        auto iter = find(U.begin(), U.end(), B[i]);
        if (iter == U.end()) {
            U.push_back(B[i]);
        }
    }

    // intersection : A * B
    for (int i = 0; i < A.size(); i++) {
        auto iter = find(B.begin(), B.end(), A[i]);
        auto iter2 = find(X.begin(), X.end(), A[i]);
        if (iter != B.end() && iter2 == X.end()) {
            X.push_back(A[i]);
        }
    }
```

```
    // print A, B
    cout << "A : ";
    for (int i = 0; i < A.size(); i++) {
        cout << A[i] << " ";
    }
    cout << endl;
    cout << "B : ";
    for (int i = 0; i < B.size(); i++) {
        cout << B[i] << " ";
    }
    cout << endl;


    // print vector U
    cout << "U = { ";
    for (int i = 0; i < U.size(); i++) {
        cout << U[i] << " ";
    }
    cout << "}" << endl;
    cout << "X = { ";
    // print vector X
    for (int i = 0; i < X.size(); i++) {
        cout << X[i] << " ";
    }
    cout << "}" << endl;
}
```

3번 문제는, C++에서의 Set STL을 이용하면 더 편하게 풀 수 있을 거라고 생각했지만,

컴알 실습에서 원하는 풀이가 아닐 것이라 생각해 배열(vector)에 담아 풀었다.

교집합이 잘 실행되는지 보기 위해 random하게 숫자를 뽑을 때 1~10까지만 뽑아 봤고,

Time Complexity는 push가 O(1)이지만, Union과 Interseciton을 만드는 과정에서

O(n)이 걸리는 find를 n번씩 수행하기 때문에 O(n^2)이다.

Space Complexity는 O(n).

```
75      for (int i = 0; i < X.size(); i++) {
76          cout << X[i] << " ";
77      }
78      cout << "}" << endl;
79  }
80
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
❯ cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -o a.out  && "/Users/
jaewonlee/Desktop/PS/"/a.out
6 6
A : 6 10 4 1 3 4
B : 2 4 10 8 10 1
U = { 6 10 4 1 3 2 8 }
X = { 10 4 1 }
```