실습 7

학과 : 전자공학과 학번 : 2023104322 이름 : 현시온

- 과제는 pdf로 변환하여 제출(과제 문서 첫 줄에 학과/학번/이름 포함)
- 과제는 순서대로 작성하며, 문제와 설명을 모두 포함(형식이 맞지 않으면 감점)
- 프로그램을 작성하는 문제는 소스코드와 실행 결과를 모두 text로 붙여넣기(그림으로 포함하지 말 것)하고 코드 설명 및 결과에 대한 설명을 포함해야 함
- 문의 사항은 이메일(nize@khu.ac.kr) 또는 오픈 카톡방을 이용

1. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
void print(std::vector<double> v) {
    for (int i = 0; i < v.size(); ++i)
        std::cout << v[i] << ", ";
    std::cout << std::endl;
}
```

//double형 자료를 엘리먼트로 하는 벡터 v를 파라미터로 하는, 함수 print 정의. 정수형 변수 i가 벡터 v의 크기보다 작은 한, 다음 과정을 반복하고 i에 1씩 더한다; 벡터 v의 i번째 인덱스에 위치한 엘리먼트와 문자열 , 을 출력한다.

```
int main() {
    std::vector<double> v0 = { 1 }, v1(3), v2(3, 2);
```

//double형 자료를 엘리먼트로 하는 벡터 v0, v1, v2를 선언하고 v0는 1을 엘리먼트로, v1의 크기는 3으로 (디폴트 엘리먼트 값은 0이다.), v2의 크기는 3으로 함과 동시에 모든 엘리먼트를 2로 설정한다.

```
    print(v0);
    print(v1);
    print(v2);
```

//인수를 각각 v0, v1, v2로 하여 차례대로 함수 print를 호출한다.

```
}
```

실행 결과:

```
1,
0, 0, 0,
2, 2, 2,
```

2. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
void print(std::vector<double> v) {
    for (int i = 0; i < v.size(); ++i)
```

```
        std::cout << v[i] << ", ";
    std::cout << std::endl;
}
int main() {
    std::vector<double> v0 = { 1 }, v1(3), v2(3, 2);

    print(v0);
    print(v1);
    print(v2);
```
//여기까지는 1번 문제의 코드와 동일한 구조를 가지므로 설명을 생략한다.
```
    v0[0] = 10;
```
//벡터 v0의 인덱스 0에 위치한 엘리먼트에 10을 할당한다.
```
    v1.at(0) = 20;
```
//벡터 v1의 인덱스 0에 위치한 엘리먼트에 20을 할당한다.
```
    v1[v2.at(1)] = 30;
```
//벡터 v2의 인덱스 1에 위치한 엘리먼트와 같은 값을 가진 벡터 v1의 인덱스에 위치한 엘리먼트에 30을 할당한다. 즉, v1의 인덱스 2에 위치한 엘리먼트에 30을 할당한다.
```
    print(v0);
    print(v1);
}
```
//각각의 벡터의 엘리먼트의 변화를 반영하여 각각 v0, v1을 인수로 하는 함수 print를 호출한다.

실행 결과:

1,

0, 0, 0,

2, 2, 2,

10,

20, 0, 30,

3. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
void square0(std::vector<double> v) {
    for (int i = 0; i < v.size(); ++i)
        v[i] *= v[i];
}
```
//double형 자료를 엘리먼트로 하는 벡터 v를 파라미터로 하는, 함수 square0 정의. 정수형 변수 i가 벡터 v의 크기보다 작은 한, 다음 과정을 반복하고 i에 1씩 더한다; 벡터 v의 i번째 인덱스에 위치한 엘리먼트를 제곱한다.
```
void square1(std::vector<double>& v) {
```

```
    for (int i = 0; i < v.size(); ++i)
        v[i] *= v[i];
}
```

//double형 자료를 엘리먼트로 하는 벡터 레퍼런스 v를 파라미터로 하는, 함수 square1 정의. 정수형 변수 i가 벡터 레퍼런스 v의 크기보다 작은 한, 다음 과정을 반복하고 i에 1씩 더한다; 벡터 레퍼런스 v의 i번째 인덱스에 위치한 엘리먼트를 제곱한다.

```
void square2(std::vector<double>& v) {
    for (auto x : v)
        x *= x;
}
```

//double형 자료를 엘리먼트로 하는 벡터 레퍼런스 v를 파라미터로 하는, 함수 square2 정의. 변수 x에 벡터 레퍼런스 v의 엘리먼트를 하나씩 차례로 받아서 초기화할 때마다 다음 과정을 반복한다; (x의 타입은 자동으로 v의 엘리먼트의 타입인 double로 정해진다.) 변수 x를 제곱한다.

```
void square3(std::vector<double>& v) {
    for (auto& x : v)
        x *= x;
}
```

//double형 자료를 엘리먼트로 하는 벡터 레퍼런스 v를 파라미터로 하는, 함수 square3 정의. 변수 레퍼런스 x에 벡터 레퍼런스 v의 엘리먼트를 하나씩 차례로 받아서 초기화할 때마다 다음 과정을 반복한다; (x의 타입은 자동으로 v의 엘리먼트의 타입인 double로 정해진다.) 변수 레퍼런스 x를 제곱한다.

```
void print(std::vector<double> v) {
    for (int i = 0; i < v.size(); ++i)
        std::cout << v[i] << ", ";
    std::cout << std::endl;
}
```

//double형 자료를 엘리먼트로 하는 벡터 v를 파라미터로 하는, 함수 print 정의. 정수형 변수 i가 벡터 v의 크기보다 작은 한, 다음 과정을 반복하고 i에 1씩 더한다; 벡터 v의 i번째 인덱스에 위치한 엘리먼트와 문자열 , 을 출력한다.

```
int main() {
    std::vector<double> v0 = { 1, 2, 3, 4, 5 };
    std::vector<double> v1 = { 1, 2, 3, 4, 5 };
    std::vector<double> v2 = { 1, 2, 3, 4, 5 };
    std::vector<double> v3 = { 1, 2, 3, 4, 5 };
```

//엘리먼트를 1, 2, 3, 4, 5 double형 자료 5개로 하는 벡터 v0, v1, v2, v3을 선언한다.

```
    square0(v0);
    print(v0);
```

//벡터 v0를 인수로 하는 함수 square0와 print를 호출한다. 함수 square0의 경우 인수를 할당한 파라미터 벡터 v의 엘리먼트를 제곱하는 내용이므로 인수에는 아무런 영향을 주지 않기 때문에 처음 선언한 그대로 함수 print에 따라 출력된다.

```
    square1(v1);
    print(v1);
```

//벡터 v1를 인수로 하는 함수 square1와 print를 호출한다. 함수 square1의 경우 v가 입력한 인수의 레퍼런스가 되고, 그 레퍼런스의 엘리먼트를 제곱하는 내용이므로 인수에도 영향을 준다. 그러므로 엘리먼트가 제곱된 v1이 함수 print에 따라 출력된다.

```
    square2(v2);
    print(v2);
```

//벡터 v2를 인수로 하는 함수 square2와 print를 호출한다. 함수 square2의 경우 v가 입력한 인수 v2의 레퍼런스가 되고, 그 레퍼런스의 엘리먼트를 새로운 변수 x에 차례대로 할당하면서 그 x를 제곱하는 내용이다. x의 변화는 v에 변화를 주지 않기 때문에 레퍼런스 v의 대상인 인수 v2에도 영향이 없다. 그러므로 처음 선언한 그대로 v2가 함수 print에 따라 출력된다.

```
    square3(v3);
    print(v3);
}
```

//벡터 v3를 인수로 하는 함수 square3와 print를 호출한다. 함수 square3의 경우 v가 입력한 인수 v3의 레퍼런스가 되고, x는 v의 엘리먼트의 레퍼런스가 되며, 그 x를 제곱하는 내용이다. 즉 x의 변화는 v에 변화를 주고, 레퍼런스 v의 대상인 인수 v2에도 영향이 간다. 그러므로 엘리먼트가 제곱된 v3가 함수 print에 따라 출력된다.

실행 결과:

```
1, 2, 3, 4, 5,
1, 4, 9, 16, 25,
1, 2, 3, 4, 5,
1, 4, 9, 16, 25,
```

4. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
std::vector<double> square(std::vector<double> v) {
    std::vector<double> r;
    for (auto x : v)
        r.push_back(x*x);

    return r;
}
```

//타입이 double인 엘리먼트를 가지는 벡터 v를 파라미터로 가지고, 타입이 double이 엘리먼트를 가지는 벡터를 반환하는 함수 square를 정의한다. 타입이 double인 엘리먼트를 가지는 벡터 r을 선언하고, 변수 x에 벡터 v의 엘리먼트를 하나씩 차례로 받아서 초기화할 때마다 다음 과정을 반복한다; (x의 타입은 자동으로 v의 엘리먼트의 타입인 double로 정해진다.) 벡터 r에 x의 제곱을 엘리먼트로 하여 추가한다. 이후 반복을 종료하고 나서, 벡터 r을 반환한다.

```
void print(std::vector<double> v) {
    for (int i = 0; i < v.size(); ++i)
        std::cout << v[i] << ", ";
    std::cout << std::endl;
}
```

//double형 자료를 엘리먼트로 하는 벡터 v를 파라미터로 하는, 함수 print 정의. 정수형 변수 i가 벡터 v의 크기보다 작은 한, 다음 과정을 반복하고 i에 1씩 더한다; 벡터 v의 i번째 인

덱스에 위치한 엘리먼트와 문자열 , 을 출력한다.

```
int main() {
    std::vector<double> v0 = { 1, 2, 3, 4, 5 }, v1 = { 100, 200 };
```

//타입이 double인 엘리먼트를 가지는 벡터 v0, v1을 선언한다. v0는 1, 2, 3, 4, 5를 엘리먼트로, v1은 100, 200을 엘리먼트로 가진다.

```
    print(v0);
    print(v1);
```

//v0, v1을 각각 인수로 하여 함수 print를 호출하여 출력한다.

```
    v1 = square(v0);
    print(v0);
    print(v1);
}
```

//v0를 인수로 하여 함수 square를 호출하여 반환된 결과를 v1에 할당한다. 함수 내용에 따르면 v1에는 v0의 원래 엘리먼트의 제곱을 엘리먼트로 하는 벡터 r이 할당된다. 또한 함수 square는 인수가 할당된 v로 함수 내용을 처리하므로 인수에는 아무런 영향이 없다. 따라서 아무런 변화가 없는 v0, 새롭게 할당된 v1을 각각 인수로 하여 함수 print를 호출하여 출력한다.

실행 결과:

```
1, 2, 3, 4, 5,
100, 200,
1, 2, 3, 4, 5,
1, 4, 9, 16, 25,
```

5. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
void print(std::vector <std::vector<double>> v) {
    for (int i = 0; i < v.size(); ++i) {
        for (int j = 0; j < v[i].size(); ++j)
            std::cout << v[i][j] << ", ";
        std::cout << ": " << v[i].size();
        std::cout << std::endl;
    }
    std::cout << std::endl;
}
```

//타입이 double인 엘리먼트를 가지는 벡터를 엘리먼트로 하는 벡터 v를 파라미터로 하는 함수 print를 정의한다. 정수형 변수 i가 벡터 v의 크기보다 작은 한, 다음 과정을 반복할 때마다 i에 1씩 더한다;

//정수형 변수 j가 인덱스 i에 위치한 벡터 v의 엘리먼트의 크기보다 작은 한, 다음 과정을 반복할 때마다 j에 1씩 더한다; 벡터 v의 인덱스 i에 위치한 엘리먼트인 벡터 v[i]의 인덱스 j에 위치한 엘리먼트 v[i][j]와 문자열 , 을 출력한다. 해당 반복을 끝내고 문자열 : 와 벡터 v[i]의 크기를 출력한다.

```
int main() {
```

```
    std::vector <std::vector<double>> v0(2, std::vector<double>(3, 1));
```

//타입이 double인 엘리먼트를 가지는 벡터를 엘리먼트로 하는 벡터 v0를 선언하고, 크기가 3이고 1을 엘리먼트로 가지는 벡터를 엘리먼트로 하며, 크기는 2로 한다.

```
    print(v0);
```

//v0를 인수로 하여 함수 print를 출력한다. 함수 내용에 따르면 벡터의 한 행을 출력할 때마다 그 행의 요소의 개수도 출력할 것이다.

```
}
```

실행 결과:

```
1, 1, 1, : 3
1, 1, 1, : 3
```

6. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
void print(std::vector <std::vector<double>> v) {
    for (int i = 0; i < v.size(); ++i) {
        for (int j = 0; j < v[i].size(); ++j)
            std::cout << v[i][j] << ", ";
        std::cout << ": " << v[i].size();
        std::cout << std::endl;
    }
    std::cout << std::endl;
}
```

//문제 5번의 print 함수와 동일한 내용이므로 설명은 생략한다.

```
int main() {
    std::vector <std::vector<double>> v0 = { { 1 }, {2, 3}, {4, 5} };
```

//타입이 double인 엘리먼트를 가지는 벡터를 엘리먼트로 하는 벡터 v0를 선언하고, v0는 1을 엘리먼트로 가지는 벡터, 2, 3을 엘리먼트로 가지는 벡터, 4, 5를 엘리먼트로 가지는 벡터를 엘리먼트로 가진다.

```
    print(v0);
}
```

//v0를 인수로 하여 함수 print를 호출한다. v0의 엘리먼트인 벡터들은 열에 해당할 것으로 예상할 수 있다.

실행 결과:

```
1, : 1
2, 3, : 2
4, 5, : 2
```

7. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
void print(std::vector<int> v) {
    for (int i = 0; i < v.size(); ++i)
        std::cout << v[i] << ", ";
    std::cout << std::endl;
}
//벡터를 출력하기 위한 함수 print를 정의한다.
int main() {
    std::vector<int> v;

    srand(static_cast<unsigned>(time(0)));
    for (int i = 0; i < 10; i++)
        v.push_back(rand() % 100-50);
    print(v);
    //벡터 v에 -50~50 범위의 무작위 값을 받아 요소로 채워 출력한다.
    for (int i = 0; i < v.size(); ++i) {
        int t = v[i];
        v[i] = v[v.size() - i - 1];
        v[v.size() - i - 1] = t;
    }
    print(v);
//벡터 v의 대칭되는 요소들을 서로 뒤바꾼다. 하지만 모든 자리에 대해서 뒤바꾼다면 결국 같은 결과를 얻을 뿐이다.
    for (int i = 0; i < v.size()/2; ++i) {
        int t = v[i];
        v[i] = v[v.size() - i - 1];
        v[v.size() - i - 1] = t;
    }
    print(v);
}
//하지만 벡터의 v의 요소를 절반 범위에 대해서만 뒤바꾼다면 요소가 반대로 뒤집힌 벡터 v를 얻을 수 있다.
```

8. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
void print(std::vector<int> v) {
    for (int i = 0; i < v.size(); ++i)
        std::cout << v[i] << ", ";
    std::cout << std::endl;
}
```

//double형 자료를 엘리먼트로 하는 벡터 v를 파라미터로 하는, 함수 print 정의. 정수형 변수 i가 벡터 v의 크기보다 작은 한, 다음 과정을 반복하고 i에 1씩 더한다; 벡터 v의 i번째 인덱스에 위치한 엘리먼트와 문자열 , 을 출력한다.

```
int main() {
    std::vector<int> v;
```

//정수형 자료를 엘리먼트로 하는 벡터 v를 선언한다.

```
    srand(static_cast<unsigned>(time(0)));
```

//현재 시간에 따라 난수의 생성 기준을 다르게 한다.

```
    for (int i = 0; i < 10; i++)
        v.push_back(rand() % 10);
```

//정수형 변수 i가 10보다 작은 한, 다음 과정을 반복하며 i에 1씩 더한다; 생성한 난수의 10을 나눈 결과의 나머지를 벡터 v에 엘리먼트로 추가한다.

```
    print(v);
    v.clear();
```

//벡터 v를 출력하고, 벡터 v의 모든 엘리먼트를 제거한다.

```
    for (int i = 0; i < 10; i++) {
        int k, r;
        do {
            r = rand() % 10;
            for (k = 0; k < i; k++)
                if (r == v[k]) break;
        } while (k < i);
        v.push_back(r);
    }
```

//정수형 변수 i가 10보다 작은 한, 다음 과정을 반복하며 i에 1씩 더한다; 정수형 변수 k, r을 선언하고, 일단 생성한 난수의 10을 나눈 결과의 나머지를 r에 할당한 다음, 만약 r과 같은 v의 엘리먼트가 있다면 for에서 바로 나와 do-while문의 반복을 무조건 다시 하게 되어 있다. Break로 for문을 빠져나온다면 k가 i보다 무조건 작기 때문에 do-while문의 조건을 만족하기 때문이다. 즉 v의 엘리먼트와 다른 r이어야지 무사히 k를 i만큼 증가시킬 수가 있고, 이를 통해 do-while문에서 무사히 빠져나올 수 있다. 그리고 이렇게 험난한 반복 과정을 빠져나온 r을 벡터 v에 엘리먼트로 추가한다.

```
    print(v);
}
```

//만들어진 벡터 v를 인수로 하여 함수 print를 호출하고 결과를 출력한다. 벡터 v의 엘리먼트들은 겹치지 않을 것임을 예측할 수 있다.

실행 결과:

```
6, 7, 2, 9, 5, 3, 6, 1, 6, 5,
9, 8, 3, 0, 5, 6, 2, 7, 1, 4,
```

9. 다음과 같은 동작은 수행하는 코드를 작성하라.

   A. 크기가 5인 정수(int)를 저장하는 vector를 선언(v1),

   B. v1의 모든 요소에 [0, 99] 범위의 임의의 값을 저장,

   C. `std::vector<int> v2 = Reverse(v1);`의 코드로 v2에 v1의 요소가 역순으로 저장될 수 있는 Reverse 함수 정의, (예, v1의 요소가 {4, 6, 1, 3, 2}이면, v2의 요소는 {2, 3, 1, 6, 4}로 저장)

   D. v1과 v2를 화면에 출력

```
#include <iostream>
#include <vector>
#include <ctime>
#include <cstdlib>

void print(std::vector<int> v) {
	for (int i = 0; i < v.size(); ++i) {
		std::cout << v[i] << ", ";
	}
	std::cout << std::endl;
}
//벡터를 출력하기 위한 함수 print 정의
std::vector<int> Reverse(std::vector<int> v) {
	std::vector<int> reverse(v.size());
	for (int i = 0; i < v.size(); ++i) {
		reverse[i] = v[v.size() - i - 1];
	}
	return reverse;
}
//벡터의 요소를 뒤집는 함수 Reverse를 구현하기 위해 새로운 벡터의 인덱스를 v.size() - i -1로 표현.
int main() {
	srand(static_cast<unsigned>(time(0)));
	std::vector<int> v1(5);
	for (int i = 0; i < 5; ++i) {
		v1[i] = rand() % 100;
	}
	std::vector<int> v2 = Reverse(v1);
	print(v1);
	print(v2);
}
```

//매시간마다 생성 방식이 변화하는 난수를 통해 0~99 범위의 무작위 정수 5개를 v1의 요소로 할당. 이후 v1의 요소를 뒤집은 벡터 v2를 함수 Reverse를 통해 구현하고 함수 print를 통해

이 둘을 출력.

10. 다음과 같은 동작을 수행하는 코드를 작성하라.

A. 크기가 3x2인 정수(int)를 저장하는 2차원 vector를 선언(m1), (m1의 요소는 [0][0]에서 [2][1]까지 사용할 수 있는 형식)

B. m1의 모든 요소에 [0,99] 범위의 임의의 값을 저장,

C. `std::vector<std::vector<int>> m2 = Transpose(m1);`의 코드로 m2의 [i][k]의 요소가 m1의 [k][i]요소로 저장될 수 있는 Transpose 함수 정의

D. m1과 m2를 화면에 출력

```
#include <iostream>
#include <vector>
#include <ctime>
#include <cstdlib>

void print(std::vector <std::vector<int>> v) {
	for (int i = 0; i < v.size(); ++i) {
		for (int j = 0; j < v[i].size(); ++j)
			std::cout << v[i][j] << ", ";
		std::cout << std::endl;
	}
	std::cout << std::endl;
}
//벡터를 출력하기 위한 함수 print 정의
std::vector<std::vector<int>> Transpose(std::vector<std::vector<int>> v) {
	std::vector<std::vector<int>> trans(2, std::vector<int>(3));
	for (int i = 0; i < 2; ++i) {
		for (int j = 0; j < 3; ++j) {
			trans[i][j] = v[j][i];
		}
	}
	return trans;
}
//벡터의 요소를 뒤집는 함수 Transpose 정의
int main() {
	srand(static_cast<unsigned>(time(0)));
	std::vector<std::vector<int>> m1(3, std::vector<int>(2));
	for (int i = 0; i < 3; ++i) {
		for (int j = 0; j < 2; ++j) {
			m1[i][j] = rand() % 100;
		}
	}
	std::vector<std::vector<int>> m2 = Transpose(m1);
	print(m1);
	print(m2);
}
//벡터 m1의 요소를 0~99 무작위로 받고 함수 Transpose를 통해 벡터 m1의 요소를 뒤집은 벡터 m2 생성 및 함수 print를 통한 m1, m2 출력
```