실습 3

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++ 코드에서 수식이 평가되는 과정과 결과를 설명하라.

```
#include <iostream>
int main() {
   int x = 7, y = 3, z;
```

//변수 x, y, z을 정수형으로 선언 및 x에 정수 7, y에 정수 3을 할당한다.

```
   z = x / y * 10;
   std::cout << z << std::endl;
```

//연산자 =, /, * 중 우선순위가 높은 /와 * 연산자 계산을 먼저 행하는데, 이 둘의 결합성을 고려하여 /, * 순서로 계산한다. 변수 x와 y는 정수형이므로 나눗셈의 결과도 정수형, 즉 7/3 나눗셈의 나머지를 제외한 몫인 2를 도출하며, 이 결과에 10을 곱하게 되어 20이라는 결과가 나온다. 이제 우선순위가 낮은 = 연산자의 역할에 따라 변수 z에 정수 20을 할당한 다음, z를 출력한다.

```
   z = x + y / 2;
   std::cout << z << std::endl;
```

//연산자 =, +(Binary), / 중 우선순위가 높은 / 연산자 계산을 먼저 행하는데, 정수형 사이의 나눗셈이므로 그 결과도 정수형, 즉 3/2 나눗셈의 나머지를 제외한 몫인 1을 도출한다. 이제 그 다음으로 우선순위가 높은 + 연산자에 따라 7과 1을 더하게 되어 8이라는 결과가 나온다. 이제 우선순위가 가장 낮은 = 연산자의 역할에 따라 변수 z에 정수 8을 할당한 다음, z를 출력한다.

```
}
```

코드 실행 결과:

```
20
8
```

2. 아래 C++ 코드에서 수식이 평가되는 과정과 결과를 설명하라.

```
#include <iostream>
int main() {
   int x = 7, y = 3, z1;
   double z2;
```

//변수 x, y, z1을 정수형으로 선언 및 x에 정수 7, y에 정수 3을 할당하고, 변수 z2는 실수형으로 선언한다.

```
   z1 = static_cast<double>(x) / y * 10;
```

//연산자 =, /, * 중 우선순위가 높은 /와 * 연산자 계산을 먼저 행하는데, 이 둘의 결합성을

고려하여 /, * 순서로 계산한다. 변수 x는 원래 정수형이지만, 해당 수식에서는 실수형으로 casting해주었기 때문에 실수형으로 취급한다. 즉 실수형과 정수형의 나눗셈이므로 결과는 실수형으로 도출되어야 하기 때문에 나눗셈의 결과는 그대로 7/3이며, 이 결과에 10을 곱하게 되어 70/3이라는 결과가 나온다. 이제 우선순위가 낮은 = 연산자의 역할에 따라 변수 z1에 70/3을 할당하게 되는데, z1는 정수형이기 때문에 70/3의 소수점 자리를 제외한 23으로 변환되어 할당 및 출력된다.

```
   z2 = static_cast<double>(x) / y * 10;
```

//연산자 =, /, * 중 우선순위가 높은 /와 * 연산자 계산을 먼저 행하는데, 이 둘의 결합성을 고려하여 /, * 순서로 계산한다. 변수 x는 원래 정수형이지만, 해당 수식에서는 실수형으로 casting해주었기 때문에 실수형으로 취급한다. 즉 실수형과 정수형의 나눗셈이므로 결과는 실수형으로 도출되어야 하기 때문에 나눗셈의 결과는 그대로 7/3이며, 이 결과에 10을 곱하게 되어 70/3이라는 결과가 나온다. 이제 우선순위가 낮은 = 연산자의 역할에 따라 변수 z2에 70/3을 double 타입이 구현 가능한 숫자 범위 내에서 소수로 변환되어 할당 및 출력된다.

```
   std::cout << z1 << " " << z2 << std::endl;


   z1 = x + static_cast<double>(y) / 2;
```

// 연산자 =, +(Binary), / 중 우선순위가 높은 / 연산자 계산을 먼저 행하는데, 실수형으로 casting된 y와 정수 2 사이의 나눗셈이므로 그 결과는 실수형, 즉 3/2의 결과 1.5를 도출한다. 이제 그 다음으로 우선순위가 높은 + 연산자에 따라 7과 1.5을 더하게 되어 8.5이라는 결과가 나온다.(정수형과 실수형의 덧셈의 결과 역시 실수형이다.) 이제 우선순위가 가장 낮은 = 연산자의 역할에 따라 변수 z1에 8.5를 할당하게 되는데, z1은 정수형이기 때문에 소수점 자리를 제외한 8이 할당 및 출력된다.

```
   z2 = x + static_cast<double>(y) / 2;
```

// 연산자 =, +(Binary), / 중 우선순위가 높은 / 연산자 계산을 먼저 행하는데, 실수형으로 casting된 y와 정수 2 사이의 나눗셈이므로 그 결과는 실수형, 즉 3/2의 결과 1.5를 도출한다. 이제 그 다음으로 우선순위가 높은 + 연산자에 따라 7과 1.5을 더하게 되어 8.5이라는 결과가 나온다.(정수형과 실수형의 덧셈의 결과 역시 실수형이다.) 이제 우선순위가 가장 낮은 = 연산자의 역할에 따라 실수형인 변수 z1에 그대로 8.5가 할당 및 출력된다.

```
   std::cout << z1 << " " << z2 << std::endl;
}
```

코드 실행 결과:

```
23 23.3333
8 8.5
```

3. 아래 C++ 코드에서 zero의 출력이 0이 아닌 이유에 대해서 설명하라.

```
#include <iostream>
int main() {
    double one = 1.0, one_fifth = 1.0 / 5.0,
        zero = one - one_fifth - one_fifth - one_fifth - one_fifth
        - one_fifth;
    std::cout << "one = " << one << ", one_fifth = " << one_fifth
        << ", zero = " << zero << '\n';
}
```

이유: 컴퓨터에서는 소수점 자릿수를 표현할 때도 마찬가지로 이진수를 통해 구현해야 한다.(부동 소수점 형태) 하지만 이진수의 한계로 인해 구현 과정에서 어쩔 수 없이 약간의 오차가 생기게 되며, 해당 코드에서 zero의 출력이 0이 아닌 것이 그것을 증명한다.

4. 아래의 C++ 코드의 출력이 127, -128인 이유를 설명 설명하라.

```
#include <iostream>

int main() {

    char ch1 = 'A', ch2;

    ch2 = ch1 + 62;

    std::cout << static_cast<int>(ch2) << std::endl;

    ch2 = ch1 + 63;

    std::cout << static_cast<int>(ch2) << std::endl;

}
```

이유: 타입 char은 기본 8bit를 할당 받는 signed 형태이기 때문에, -128~127 범위의 총 256(2의 8승)개의 수가 순환하는 구조를 띠고 있다. 문자형 A는 ASCII CODE에 따라 65에 대응한다. 정수형과 문자형의 덧셈의 결과는 정수형이므로 ch2는 65+62=127에 대응하는 문자를 첫번째로 할당 받은 다음, 출력할 땐 int로 casting되었으므로 그대로 127을 출력한다. 한편 다음으로는 ch2에 65+63=128에 대응하는 문자가 할당되어야 할 것 같지만, 128은 signed 8bit로 구현 가능한 이진수 자릿수를 벗어나 overflow된 숫자이므로 -128~127 범위를 순환시켜 -128에 대응하는 문자가 할당되고, 마찬가지로 출력할 때 int로 casting되었기에 그대로 -128이 출력된다.

5. 아래의 C++ 코드의 출력이 255, 0인 이유를 설명하라.

```
#include <iostream>

int main() {

    unsigned char ch1 = 100, ch2;

    ch2 = ch1 + 155;

    std::cout << static_cast<int>(ch2) << std::endl;

    ch2 = ch1 + 156;

    std::cout << static_cast<int>(ch2) << std::endl;

}
```

이유: 타입 char은 기본 8bit를 할당 받는 signed 형태이지만, unsinged 형태의 경우 음수를 제외한 0~255 범위의 총 256(2의 8승)개의 수가 순환하는 구조를 띠고 있다. ch2는 덧셈의 결과인 255에 대응하는 문자를 첫번째로 할당 받은 다음, 출력할 땐 int로 casting되었으므로 그대로 255을 출력한다. 한편 다음으로는 ch2에 덧셈의 결과인 256에 대응하는 문자가 할당되어야 할 것 같지만, 256은 unsigned 8bit로 구현 가능한 이진수 자릿수를 벗어나 overflow된 숫자이므로 0~255 범위를 순환시켜 0이라는 값에 대응하는 문자가 할당되고, 마찬가지로 출력할 때 int로 casting되었기에 그대로 0이 출력된다.

6. 아래의 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>

int main() {

    int x = 10, y = 7, z; //x, y, z를 정수형으로 선언 및 x에 10, y에 7 할당

    z = ++x * y--; //전위증가의 피연산자는 x, 후위감소의 피연산자는 y이다. 먼저 후위감소 연산자에 따라 y는 1만큼 감소한 결과인 6이 할당되지만, 해당 문장에서의 연산에서는 1만큼 감소하기 전 상태로 곱셈을 진행하게 된다. 한편 전위증가 연산자에 따라 x는 1만큼 증가하여 11이 할당되며, 해당 문장에서의 연산에서도 1만큼 증가한 상태로 곱셈을 진행한다. 그래서 결론적으로 z에는 11*7의 결과 77이 할당된다. 참고로 연산자의 우선순위는 ++&--(postfix), ++&--(prefix), *, = 이다.
```

```
    std::cout << x << std::endl; //x에 할당된 11이 출력된다.
    std::cout << y << std::endl; //y에 할당된 6이 출력된다.
    std::cout << z << std::endl; //z에 할당된 77이 출력된다.
}
```

코드 실행 결과:

```
11
6
77
```

7. 아래의 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
int main() {
    int x = 3, y = 2, t; //x, y, z를 정수형으로 선언 및 x에 3, y에 2 할당

    x = y; //x에 y에 할당된 2를 할당
    y = x; //y에 x에 할당된 2를 할당
    std::cout << x << std::endl; //x에 최종적으로 할당된 값인 2를 출력
    std::cout << y << std::endl; //y에 최종적으로 할당된 값인 2를 출력

    x = 3; //x에 3을 할당
    y = 2; //y에 2를 할당
    t = x; //t에 x에 할당된 3을 할당
    x = y; //x에 y에 할당된 2를 할당
    y = t; //y에 t에 할당된 3을 할당
    std::cout << x << std::endl; //x에 최종적으로 할당된 값인 2를 출력
    std::cout << y << std::endl; //y에 최종적으로 할당된 값인 3을 출력
}
```

코드 실행 결과:

```
2
2
2
3
```

8. 두 개의 정수를 사용자로부터 입력을 받아서, 덧셈, 뺄셈, 곱셈, 나눗셈의 몫(quotient)과 나머지(remainder)를 출력하는 프로그램을 작성하라.

```
#include <iostream>
int main() {
    int x, y;
```

```
    std::cin >> x;
    std::cin >> y;
    std::cout << "Addition -> " << x + y << std::endl;
    std::cout << "Subtraction -> " << x - y << std::endl;
    std::cout << "Multiplication -> " << x * y << std::endl;
    std::cout << "Division Quotient -> " << x / y << std::endl;
    std::cout << "Division Remainder -> " << x % y << std::endl;
}
```

9. 두 개의 실수를 사용자로부터 입력을 받아서, 덧셈, 뺄셈, 곱셈과 나눗셈의 결과를 출력하는 프로그램을 작성하라.

```
#include <iostream>
int main() {
    double x, y;
    std::cin >> x;
    std::cin >> y;
    std::cout << "Addition -> " << x + y << std::endl;
    std::cout << "Subtraction -> " << x - y << std::endl;
    std::cout << "Multiplication -> " << x * y << std::endl;
    std::cout << "Division -> " << x / y << std::endl;
}
```

10. 블록 주석을 중첩해서 사용할 수 없음을 예를 들어 설명하라.

/*은 */이 나타날 때까지 모든 내용을 주석 처리한다. 따라서 블록 주석 안에 또 다른 블록 주석이 들어간다면 각각의 블록 주석의 경계가 모호해서 오류가 발생한다.

예시: /*C++ 프로그래밍은 /*웃음이 도질 정도로*/ 재미있다.*/

컴퓨터는 /*을 보고 첫번째 */가 보일때까지 주석 처리하기에 'C++ 프로그래밍은 /*웃음이 도질 정도로'라는 내용이 전부 주석 처리되지만, ' 재미있다.' 내용과 */은 처리하지 못하고 오류에 빠지게 된다.