2008학년도 1학기 기말 평가 모범 답안

고급 객체 지향 프로그래밍

동서대학교

강대기

문제 1) C++에서 class와 struct의 차이를 서술하라.

class 는 default 가 private 이고, struct 는 default 가 public 이다.

다음에 대해서 예, 아니오로 답하라.

클래스가 클래스를 상속할 수 있음 (예)
클래스가 구조체를 상속할 수 있음 (예)
구조체가 클래스를 상속할 수 있음 (예)
구조체가 구조체를 상속할 수 있음 (예)

문제 2) 프렌드를 설정하려는 다음의 세 가지 시도에서 무엇이 잘못되었는가? 잘못된 이유를 자세히 서술하고, 올바르게 고쳐라.

(a)

```
class snap {
  friend clasp;
  ...
};
class clasp {
...
};
```

프렌드 선언이 다음과 같아야 한다.

```
class snap {
  friend class clasp;
  ...
};
```

(b)

```
class cuff {
public: void snip(muff &) { ... }
  ...
};
class muff {
  friend void cuff::snip(muff &);
  ...
};
```

컴파일러가 cuff를 해석하는 도중, void snip(muff &)의 muff를 인식하려면, 사전 선언이 필요하다.

```
class muff; // 사전 선언
class cuff {
public: void snip(muff &) { ... }
  ...
};
```

```
class muff {
  friend void cuff::snip(muff &);
  ...
};
```

(c)

```
class muff {
  friend void cuff::snip(muff &);
  ...
};
class cuff {
public: void snip(muff &) { ... }
  ...
};
```

컴파일러가 void cuff::snip(muff &)를 인식할 수 있도록, class cuff 정의를 class muff보다 앞에 둔다.
그리고, class cuff 정의에서 void snip(muff &) { ... }를 인식할 수 있도록 class muff를 사전 정의한다.

```
class muff;
class cuff {
public: void snip(muff &) { ... }
  ...
};
class muff {
  friend void cuff::snip(muff &);
  ...
};
```

문제 3) 다음은 무엇을 뜻하는가?

(a) double (*pf[3])(double, double);

pf는 함수 포인터 3 개로 이루어진 배열로, 그 함수 포인터는 double 과 double 두개를 인자로 받아서 double 을 반환함.

(b) typedef int (&rifii) (int, int);

rifii는 함수에 대한 참조를 나타내는 typedef로 정의된 별명으로, 그 함수는 두 개의 int 들을 인자로 받고 int 를 반환함.

문제 4) 클래스 템플릿은 부분 특수화가 가능하다. 그렇다면 함수 템플릿은 어떤가? 함수 템플릿도 부분 특수화가 가능한가?
만일 그렇다면 그 예를 하나 보여라.
만일 그렇지 않다면 그것을 대신한 방법이 무엇인지를 서술하라.

함수 템플릿은 부분 특수화가 안된다. 그것을 대신할 방법은 함수 템플릿의 오버로딩이다. (EC++ 항목 25)

문제 5) Singleton 패턴을 클래스 템플릿으로 구현하라.

```
template <class T>
class Singleton
{
public:
  static T* getInstance()
  {
    if (0==m_pInstance) m_pInstance = new T();
    return m_pInstance;
  }

  static void releaseInstance()
  {
    if (0!=m_pInstance) delete m_pInstance;
    m_pInstance = 0;
  }

private:
  Singleton<T>() {};
  static T* m_pInstance;
};
```

문제 6) 일단 객체를 한 번 만들면, 다른 객체로 복사되지 못하도록 하는 Uncopyable 클래스를 설계하여 구현하라.

```
class Uncopyable {

...

private:
  Uncopyable(const Uncopyable&);
  Uncopyable& operator=(const Uncopyable&);
};
```

문제 7) 다음은 1에서 100까지 출력하는 프로그램의 주된 부분이다.

```
double ar[100+1] = {1};
transform(ar,ar+100-1,ar+1,bind1st(plus<double>(),1));
ostream_iterator<double, char> out(cout," ");
copy(ar,ar+100,out);
cout << endl;
```

이 프로그램을 기초로, 다음과 같이 1에서 100까지 각각의 항에 대한 합계(sum)를 출력하도록 위의 프로그램을 변경하라.

1 3 6 10 15 21 28 36 45 55 .... 4560 4656 4753 4851 4950 5050

```
#include <iostream>
#include <algorithm>
#include <functional>
#include <iterator>
using namespace std;

const int MAXNUM = 100;

int main()
{
  double ar[MAXNUM+1] = {1};
  transform(ar,ar+MAXNUM-1,ar+1,bind1st(plus<double>(),1));
  transform(ar,ar+MAXNUM,ar+1,ar+1,plus<double>());
  ostream_iterator<double, char> out(cout," ");
  copy(ar,ar+MAXNUM,out);
  cout << endl;
  return 0;
}
```

문제 8) 앞의 문제의 프로그램을 참조하여, transform 을 단 한번만 사용하여, 다음과 같이 피보나치 급수를 30 항 구하는 프로그램의 주된 부분을 작성하라.
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040

```
#include <iostream>
#include <algorithm>
#include <functional>
#include <iterator>

using namespace std;

const int MAXNUM = 30;

int main()
{
  double ar[MAXNUM+ 1] = {1, 1};
  transform(ar,ar+ MAXNUM-2,ar+ 1,ar+ 2,plus<double>());
  ostream_iterator<double, char> out(cout," ");
  copy(ar,ar+ MAXNUM,out);
  cout << endl;
  return 0;
}
```

또는

```
#include <algorithm>
#include <functional>
#include <iostream>
#include <iterator>
#include <vector>

using namespace std;

int main()
{
```

```
    int N = 0;
    cin >> N;
    vector<int> ar(N,1);
    transform(ar.begin(),ar.end()-2,ar.begin()+ 1,ar.begin()+ 2,plus<double>());
    ostream_iterator<double, char> out(cout," ");
    copy(ar.begin(),ar.end(),out);
    cout << endl;
    return 0;
}
```

문제 9) 다음의 제한 사항들을 지키면서, 1에서 100까지의 합인 5050을 계산해서 출력하는 프로그램을 작성하라.

● for, while, do 등을 이용하여 루프를 쓸 수 없으며, if 문이나? : 도 쓸 수 없다. 또한 switch 나 goto 문도 못 쓴다.

● 함수는 오직 main 만 있어야 하며, main 함수는 어떠한 전달 인자를 받아서도 안 되며, 정적 변수를 사용할 수도 없다.

● C++ 배열을 포함한 STL 컨테이너들을 사용할 수 없다.

● Summation을 구하는 공식을 사용할 수 없다.

```
#include <iostream>

using namespace std;

template<int X> struct SumNumber
{
    enum { sumofNumber = X + SumNumber<X-1>::sumofNumber};
};

template<> struct SumNumber<0>
{
    enum { sumofNumber = 0};
};

int main(int argc, char* argv[])
{
  cout << SumNumber<100>::sumofNumber << endl;
  return 0;
}
```

참고: switch 나 goto 문을 사용한 경우나, (101*100)/2를 사용한 경우는 이번엔 인정하였음.

문제 10). 다음과 같은 valarray 연산이 있다고 하자. 이 연산의 vector-STL 버전을 쓰라.

vad3 = 10.0 * ((vad1+ vad2)/2.0 + vad1*cos(vad2));

```
transform(vec1.begin(), vec1.end(), vec2.begin(), vec4.begin(), plus<double>());
transform(vec4.begin(), vec4.end(), vec4.begin(), bind2nd(divides<double>(),2.0));
transform(vec2.begin(), vec2.end(), vec5.begin(), cos);
transform(vec1.begin(), vec1.end(), vec5.begin(), vec5.begin(), multiplies<double>());
transform(vec4.begin(), vec4.end(), vec5.begin(), vec4.begin(), plus<double>());
transform(vec4.begin(), vec4.end(), vec4.begin(), bind2nd(multiplies<double>(),10.0));
```

또는

```
double func(double v1, double v2) { return 10.0*((v1+ v2)/2.0+ v1*cos(v2)); }
...
transform(vec1.begin(), vec1.end(), vec2.begin(), vec4.begin(), func);
```