

SSE2034

Fall 2021

Professor: Seokin Hong

TA: Donghun Jeong, Sang Jae Park, Songyu Seok

December 6, 2021

## FINAL EXAM

NAME: \_\_\_\_\_ SUNET ID: \_\_\_\_\_

Problem #	1	2	3	Total
Score				
Possible	30	40	30	100

### Instructions:

- 시험 시간은 총 3 시간이며, 점수의 총 합은 100 점입니다. (18:15 ~ 21:15)
- 문항은 총 3 문항이며 1, 2, 3-1 은 코드를 작성하는 문항이고, 3-2 는 답변을 answer sheet 에 작성하여야 합니다.
- 전자자료를 포함한 모든 자료의 활용이 가능합니다. 하지만 다른 사람과 통신은 금지됩니다.
- 코로나 19 상황으로 온라인 시험으로 진행되지만, 응시 환경(집, 컴퓨터)는 녹화됩니다.
- **부정행위가 적발될 경우 F 학점이 부여됩니다.**
- 시험 문제와 관련된 질문이 있으면 구글스프레드시트(<https://url.kr/kq3p51>)에 질문하세요.
- 문제에서 별도의 언급이 없다면 이미 포함된 라이브러리 이외의 라이브러리 사용은 불가합니다.
- 코드 파일을 final\_[student\_id].zip 으로 압축하여 icampus 에 21 시 15 분까지 제출하세요.
- Answer sheet 워드 파일을 final\_[student\_id].docx 로 icampus 에 21 시 15 분까지 제출하세요.

### Initial Setting & Submission Guideline:

- git clone <https://github.com/SSE2034/FINAL>
- sh init.sh #Initial Setting
- sh submission.sh [student\_id] #Generate Submission Zip File

1. **MATLAB [30pts]**. MATLAB program 은 행렬을 기반으로 하는 program 으로서 여러 연구에 사용되고 있다. 본 문제에서는 C++ class 를 활용하여 MATLAB 과 같이 행렬 연산을 지원하도록 한다. 1.1~1.5 는 integer matrix 를 사용하고, 1.6 에서는 template 을 활용하여 integer 와 float 을 모두 지원하도록 구현하여야.

- 1.1 Encapsulation(3)
  - 행렬 연산을 지원하도록 하는 class 를 ADT(Abstract data type) form 으로 구현하도록 한다.
- 1.2 Constructor & Destructor(4)
  - Constructor : Matrix size 를 나타내는 row\_num, col\_num 을 인자값으로 받는다. 이때, matrix 는 dynamic array 를 통해 생성한다. 모든 값을 0 으로 초기화해준다.
  - Copy constructor : Matrix 를 인자로 받아, 같은 size, value 를 갖는 matrix 를 생성한다.
  - Destructor : 생성한 Dynamic array 를 소멸시키도록 한다.
- 1.3 Mutator & Accessor function (2)
  - row, col 값을 통해 matrix 의 value 를 set/get 하도록 한다.
- 1.4 Matrix Printing function (1)
  - Matrix 를 2 차원으로 print 하도록 한다.
- 1.5 operator overloading(10)
  - +, -, \* 연산을 두 가지 방식으로 overloading 한다.
  - 아래의 문장에서, @는 +, -, \* 중 하나의 연산을 나타낸다.
  - 첫 번째 방식은, matrix @ value 로써, 모든 원소에 대해 각각 @ 연산을 수행한다.
  - 두 번째 방식은, matrix @ matrix 로써, 일반적인 두 행렬간의 연산을 수행한다.
  - matrix 간의 연산이 불가능 할 경우는 test 하지 않는다.
  - Assignment operator(=) overloading 을 구현하여야. 이때, 동일한 size 만 assign 된다고 가정한다.
- 1.6 template (10)
  - template 를 활용하여 float type 연산을 지원하도록 구현하여야.
  - Float test 시 조건
    - 소수점 한 자리 수가 test 되고, 최대 2 자리까지 표현하도록 한다.
    - (1.00, 1.0, 1 과 같이 다른 표현 형식 모두 값이 맞으면 정답으로 인정)

## Skeleton Code Documentation

### ❖ Main.cc

- ◆ 제공되는 main file 은 template 적용 전 test format 으로써, template 을 test 하기 위해서는 직접 main.cc 를 변경하도록 한다.
- ◆ Template 채점시, 제공된 main 함수에서 적절하게 atoi->atof / Matrix ->Matrix<type>로 변경하여 compile 하여 채점된다.

### ❖ Make

- ◆ Make 시, 3 가지 binary file 이 나온다.
- ◆ problem1\_mm\_\*.out
  - Matrix1 @ Matrix2 연산을 나타낸다.
  - addsub 의 경우 +, - 연산을 나타낸다.
  - Mult 의 경우 \* 연산을 나타낸다.
- ◆ problem1\_mv.out
  - Matrix @ Value 연산을 나타낸다.

### ❖ Input

- ◆ Binary file 에 따라 두 가지 방식의 input 이 존재한다.
- ◆ problem1\_mm\_\*.out
  - Matrix1 과 Matrix2 의 row,col 개수가 각각 순차적으로 주어지고, matrix1 과 matrix2 의 원소의 값들이 순차적으로 주어진다.
- ◆ problem1\_mv.out
  - Matrix1 의 row, col 개수가 주어지고, matrix1 의 원소 값들이 순차적으로 주어진다. 마지막으로, value 값이 주어진다.

## Example input / output

◆ Input 에서 enter 로 표시된 값은 편의를 위한 값으로 실제 input 은 space 이다.

◆ Integer

./problem1_mm_addsub.out 2 3 2 3 1 2 3 4 5 6 7 5 3 1 6 8	./problem1_mm_mult.out 2 2 2 3 4 5 6 7 3 2 4 1 4 6	./problem1_mv.out 3 3 1 5 9 7 5 3 4 6 2 2
<pre>=====matrix1===== 1 2 3 4 5 6 =====matrix2===== 7 5 3 1 6 8 =====result_add===== 8 7 6 5 11 14 =====result_sub===== -6 -3 0 3 -1 -2</pre>	<pre>=====matrix1===== 4 5 6 7 =====matrix2===== 3 2 4 1 4 6 =====result_mul===== 17 28 46 25 40 66</pre>	<pre>=====matrix1===== 1 5 9 7 5 3 4 6 2 =====value===== 2 =====result_add===== 3 7 11 9 7 5 6 8 4 =====result_sub===== -1 3 7 5 3 1 2 4 0 =====result_mul===== 2 10 18 14 10 6 8 12 4</pre>

◆ Float

./problem1_mm_addsub.out 2 2 2 2 7.4 3.3 2.7 8.1 4.7 9.5 1.1 4.3	./problem1_mm_mult.out 2 2 2 2 1 3.3 2 8 4.7 9 1.1 4	./problem1_mv.out 2 2 1.3 2.7 4.7 3.3 1.3
<pre>=====matrix1===== 7.4 3.3 2.7 8.1 =====matrix2===== 4.7 9.5 1.1 4.3 =====result_add===== 12.1 12.8 3.8 12.4 =====result_sub===== 2.7 -6.2 1.6 3.8</pre>	<pre>=====matrix1===== 1 3.3 2 8 =====matrix2===== 4.7 9 1.1 4 =====result_mul===== 8.33 22.2 18.2 50</pre>	<pre>=====matrix1===== 1.3 2.7 4.7 3.3 =====value===== 1.3 =====result_add===== 2.6 4 6 4.6 =====result_sub===== 0 1.4 3.4 2 =====result_mul===== 1.69 3.51 6.11 4.29</pre>

2. **LRU Data Structure [40pt]** 일반적으로 메모리의 접근 속도는 연산 속도에 비해 느린 편이다. 좀더 빠른 메모리 접근을 위해 더 빠른 속도를 가진 cache 를 모든 프로세서에서 사용하고 있다. 하지만 Cache 는 용량에 한계가 있기 때문에 자신의 용량을 넘어서는 접근이 오면 하나를 선택하여 Evict 한다. 이 때 앞으로 사용하지 않을 data 를 evict 하는 것이 성능에 도움이 된다.

이를 위해 고안된 가장 기본적인 알고리즘이 LRU 라고 불리는 Least Recently Used 알고리즘이다. 가장 오랫동안 사용되지 않은 data 는 앞으로도 사용될 확률이 낮다고 생각하고 해당 data 를 evict 하는 알고리즘이다. LRU 의 기능을 하는 자료구조는 수업 시간에 배운 Priority Queue 와 같이 특수한 기능을 가진 Queue 의 형태로 표현이 가능하다.

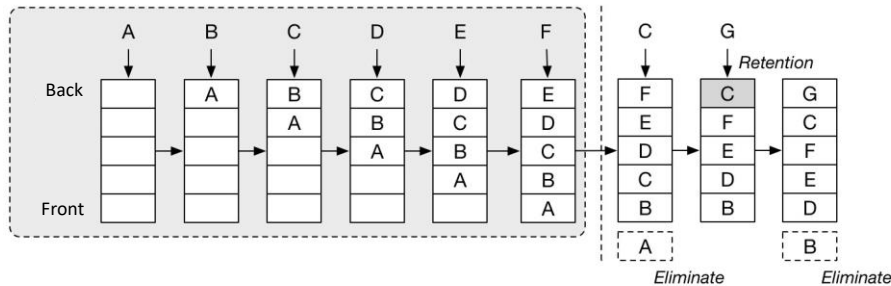


Figure 1. LRU Data Structure

Figure 1 과 같이 Front 와 Back 을 가진 Queue 에서 data 가 Queue 에 들어간다. 만약 capacity 를 초과하는 경우 Front 에 있는 값이 evict 된다. 그리고 자료구조 내부에 있는 data 를 접근하게 되면 해당 값은 Back 으로 이동하게 된다.

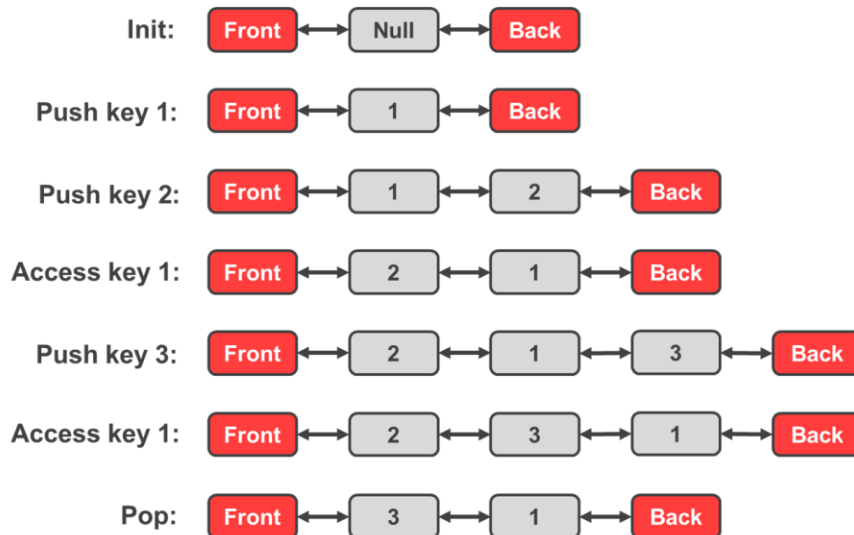


Figure 2. LRU Implementation Using Bi-Directional Linked List

해당 자료구조를 구현하기 위해 Figure 2 와 같이 Bi-Directional Linked List(양방향 linked list)를 이용할 수 있다. 해당 자료구조는 자신의 Node 이전과 다음의 pointer 를 모두 가져 List 내부의 요소를 쉽게 제거하는 데에 도움이 된다. Node 는 previous pointer, key, value, next pointer 의 요소를 가진다.

본 문제에서는 LRU 자료구조와 Cache 동작을 modeling 하는 코드의 일부분을 작성한다. Problem2/src 폴더에 skeleton code 가 주어져 있다. (코드의 실행 방법은 Problem2/Readme.md 를 참고하세요.)

- common.h : 모든 파일에서 사용하는 기본적인 PARAMETER 들이 선언되어 있다. CACHELINE 은 Cache 가 가지는 Byte Array 를 modeling 한 class 로 단순히 cache line size 만큼의 vector member variable 만 가지고 있다.
- lru.h : LRU 자료 구조를 위한 class 인 LRUNode 와 LRU Class 의 선언부분이 있는 파일이다.
- lru.cc : LRUNode 와 LRU Class 의 정의부분이 있는 파일이다.
- memory.h : cache 와 memory 구현을 위한 class 의 선언 부분이 있는 파일이다.
- memory.cc : Memory class 의 정의 부분이 있는 파일이다.
- cache.cc : Cache class 의 정의 부분이 있는 파일이다.

### Problem 2-1.

memory.h 파일을 보면 Cache class 가 Memory Class 멤버 변수를 가지고 있다. Memory class 의 데이터는 중요하기 때문에 constructor 를 제외한 모든 멤버 변수와 함수가 private 권한으로 선언되어 있다. Cache class 에서는 Memory class 의 멤버들에 대해 접근이 필요하고 Memory Class 는 Cache 에게만 데이터 접근을 허가해주고 싶다.

이를 위해 line 13 의 //TODO 부분에 한 줄의 코드를 추가해야 한다. 해당 부분의 코드를 완성하세요. (Hint : Week7 Practice) [3pt, 부분점수 없음]

### Problem 2-2.

common.h 파일은 여러 헤더파일에서 사용된다. 현재 제공된 파일을 그대로 include 하여 compile 을 하면 중복 선언(re-declaration) 에러가 발생한다.

이를 위해 7 주차 실습시간에 배운 것과 같이 전처리를 통해 compile 과정에서 해당 문제가 발생하지 않도록 해야 한다. 이를 위한 코드 3 줄(파일의 가장 처음 2 줄 + 가장 마지막 1 줄)을 추가하여 코드를 완성하세요. [3pt, 부분점수 없음]

**Problem 2-3.**

cache.cc 파일을 보면 Cache 에 쓰기 동작을 modeling 하는 write() 함수가 구현되어 있지 않다. 해당 함수를 구현하여 코드를 완성하세요. call-by-reference 로 받아온 time 변수에 cache hit 인 경우 HIT\_LATENCY 를 miss 인 경우 MISS\_LATENCY 를 증가시킨다. (Hint : 제공된 read() 함수) [10pt]

**Problem 2-4.**

LRU Class(lru.h, lru.cc)를 보면 overloading 된 2 개의 at() 함수가 존재한다.

CacheLine at(addr\_t key)의 경우 해당 key 의 Value 을 반환하고 LRU Status 를 update 한다. void at(addr\_t key, CacheLine value)는 LRU 자료구조에 존재하는 key 의 value 를 바꾸고 LRU Status 를 update 한다. at() 함수를 사용하기 전에 available()을 통해 key 가 존재하는 지 검사하므로 at() 함수의 key argument 는 항상 class 에 존재한다고 가정한다. 이 두 함수를 구현하시오. [12pt]

**Problem 2-5.**

Memory class(memory.h, memory.cc)를 보면 load() 함수와 store() 함수가 선언되어 있는 것을 확인할 수 있다. 해당 함수는 Memory 공간을 파일에서 읽어오거나 저장을 해서 결과를 중간 결과를 저장하고 이어서 시작할 수 있도록 하기 위한 함수이다.

주어진 코드에서 이미 load()함수는 구현이 완료되어 있다. 하지만 store()함수는 구현이 되어 있지 않다. 메모리에 존재하는 값들의 주소와 data 는 std::map memory\_space 에 구현되어 있다.

이 데이터를 파일로 저장하는 코드를 완성하세요. 파일의 한 line 은 주소와 data 의 나열로 구성된다. 파일의 예시는 아래와 같다. 더 자세한 예시는 Problem2/data/in.txt 를 참고하여라. [12pt]

- cache line size 가 8 인 파일의 저장 형식 : addr data0 data1 data2 data3 ... data6 data7

1000	1	2	3	4	5	6	7	8
1200	0	0	0	0	0	0	0	0
1500	10	11	12	13	14	15	16	17

### 3. Inheritance (30pts).

#### Problem 3-1. (10pts)

13 주차 상속 수업시간에 C++ STL(vector/queue/stack)을 이용한 BFS/DFS 알고리즘을 구현해 보았다. 기존 구현에서 runBfs()/runDFS()함수는 외부 함수로 정의하였다. 따라서 GraphVec 클래스의 private 멤버변수 graph 에 접근하기 위해서는 아래 연산자오버로딩과 접근자를 GraphVec 클래스에 정의 해줬어야 했다.

```
1 | int size() { return N; } //접근자
2 | std::vector<int>& operator[] (const int rhs) { return graph[rhs]; } // 연산자 오버로딩
```

하지만, 만약 C++의 Inheritance 를 잘 활용하면, 멤버 변수에 접근하기 위한 추가 코드들을 작성하지 않아도 되어, GraphVec 클래스의 코드를 간결하게 짤 수 있다. 상속을 통해 GraphVec 의 코드를 간략화 하자.

<pre>1   class GraphVec{ 2   public: 3       GraphVec(int _N) 4   : N(_N), graph(new std::vector&lt;int&gt;[_N+1]){} 5   6   public: 7       void addEdge(int _node, int* _edges, int _length); 8   9       box1 (public/protected/private) 10   11       int N; 12       std::vector&lt;int&gt;* graph; 13   }; 14   15   class SearchGraph : public GraphVec{ 16   17   public: 18       SearchGraph(int _N) : GraphVec(_N){} 19   20   public: 21       void runBFS(int _start); 22       void runDFS(int _start); 23   24   }; 25   26  </pre>	<pre>1   SearchGraph sg(9); 2   int edge1[] = {2, 3, 4}; 3   int edge2[] = {1, 3, 8}; 4   int edge3[] = {1, 2, 4}; 5   int edge4[] = {1, 3, 5}; 6   int edge5[] = {4, 6, 7, 8}; 7   int edge6[] = {5, 7}; 8   int edge7[] = {5, 6}; 9   int edge8[] = {2, 5, 9}; 10   int edge9[] = {8}; 11   12   sg.addEdge(2, edge2, 3); 13   sg.addEdge(1, edge1, 3); 14   sg.addEdge(3, edge3, 3); 15   sg.addEdge(4, edge4, 3); 16   sg.addEdge(5, edge5, 4); 17   sg.addEdge(6, edge6, 2); 18   sg.addEdge(7, edge7, 2); 19   sg.addEdge(8, edge8, 3); 20   sg.addEdge(9, edge9, 1); 21   printf("\n----- BFS ----- \n"); 22   sg.runBFS(4); 23   printf("\n----- DFS ----- \n"); 24   sg.runDFS(4);</pre>
--	---

왼쪽은 상속으로 연결된 두개의 클래스 헤더 파일이고 오른쪽은 main.cc 에서 실행되는 test1() 함수의 구현 부이다. RunDFS()/runBFS() 함수는 더 이상 외부 함수가 아닌, 자식 클래스의 멤버함수로 정의 되었다. (1) 자식클래스에서만 접근 가능도록 box1 에 올바른 선언을 선택해 코드에 채워넣어라 (2) GraphVec 에 접근자/연산자 오버로딩 없는 위 헤더파일에서 runDFS, runBFS 함수를 구현하여라. (Week13 실습 코드참고해라) 실행 결과는 실습 시간 때와 동일하다. 코드는 src/inheritance1.hh 와 src/inheritance1.cc 에서 볼 수 있다.



## Problem 3-2. (20pts)

```

1 class James{
2 public:
3     void func1(){
4         std::cout << "J func1" << std::endl;
5     }
6     virtual void func3(){
7         std::cout << "J func3" << std::endl;
8     }
9 };
10
11 class Richard : public James{
12 public:
13     virtual void func1(){
14         std::cout << "R func1" << std::endl;
15     }
16     virtual void func3(){
17         std::cout << "R func3" << std::endl;
18     }
19 };
20

```

```

1 class Donard : public Richard {
2 public:
3     virtual void func2(){
4         std::cout << "D func2" << std::endl;
5     }
6     void func4(){
7         std::cout << "D func4" << std::endl;
8     }
9 };
10
11 class Edward : public Donard{
12 public:
13     virtual void func2(){
14         std::cout << "E func2" << std::endl;
15     }
16     virtual void func4(){
17         std::cout << "E func4" << std::endl;
18     }
19 };
20

```

위 테이블은 src/inheritance2.hh 에 정의된 상속(Inheritance) 관계의 클래스(Class)들을 보여주고 있다. src/main.cc 에 각 클래스 객체(Object) 선언은 아래와 같이 정의하였다.

```

1 James* var1 = new Donard();
2 Richard* var2 = new Edward();
3 Richard* var3 = new Donard();
4 Edward* var4 = new Edward();
5 Donard* var5 = new Edward();
6 James* var6 = new Richard();

```

주어진 위 코드들로 아래의 문항들에 대해 답하여라. 코드들은 src/inheritance2.hh 에서 확인할 수 있다. main 함수에 멤버함수들을 직접 호출/실행 시키며 문제에 답하여라.

**\*\* 주의사항**

- 코드의 변경은 위 테이블에 정의된 상태를 기준으로 어떻게 바뀌어야 하는지를 뜻한다.
- 코드의 변경은 src/inheritance2.hh 에 정의된 클래스의 멤버 함수 수정/추가만을 허용한다. 허용하는 수정/추가는 아래 c,d,e 만을 허용한다. 그외 코드 수정은 일정 허용하지 않는다. 예를들어, 별개의 std::cout 문을 추가하는 것은 인정하지 않는다.
- 멤버 함수 추가시 함수의 이름은 func1/func2/func3/func4 중 하나 이다.
- 멤버 함수 변경시, 함수의 가상화(virtual) 선언을 추가/제거 할 수 있다.
- 멤버 함수 변경시, 다른 클래스의 함수를 호출 할 수 있다. 단, James::func1()과 같이 :: 연산자를 사용한 함수 호출을 불가능하다.

## Answer Example)

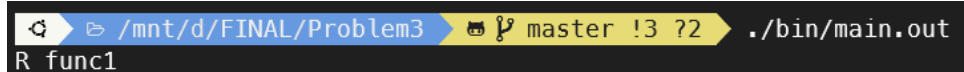
**Example Questions-1)**

main.cc 에서 var6->func1() 을 실행 시켜보고 에러가 발생하는지 답하라.

만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

**Example Questions-2)**

main.cc 에서 var6->func1() 을 실행시킬때, Richard 의 func1() 만을 호출 되게 하려면, 코드를 어떻게 변경시켜야 하는지 설명하시오. 아래는 원하는 콘솔출력이다.



```

$ ./bin/main.out
R func1

```

**Example Ans)**

오류는 발생하지 않으며, James 의 func1()이 호출된다.

James 에 정의된 func1()을 virtual type 으로 수정한다.

Or

James 클래스의 멤버변수를 아래와 같이 변경한다.

```

virtual void func1(){
    std::cout << "J func1" << std::endl;
}

```

(\* 답안 작성 가이드라인 추가 코멘트 \*)

위 예시 답안처럼 코드를 올리셔도 되고 설명만을 쓰셔도 답안으로 인정됩니다.

만약 James 클래스의 멤버변수를 아래 처럼 변경시

```

void func1(){
    std::cout << "R func1" << std::endl;
}

```

주의 사항란의 제한사항을 벗어나니 설령 콘솔 출력이 맞더라도 답안으로 인정하지 않습니다.

(Richard 의 func1 이 호출된것도 아니니 애초에 정답이 될 수 없습니다.)

**Questions 1-1)**

main.cc 에서 var1->func2() 을 실행 시켜보고 에러가 발생하는지 답하라.

만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

**Questions 1-2)**

main.cc 에서 var1->func2() 을 실행시킬 때, Donard 의 func2()만 호출 되게 하려면,

코드를 어떻게 변경시켜야 하는지 설명하시오. 아래는 원하는 콘솔 출력이다.

```
➤ /mnt/d/FINAL/Problem3 ➤ master !3 ?2 ➤ ./bin/main.out
D func2
```

Ans)

#### Questions 2-1)

main.cc 에서 var2->func2() 을 실행 시켜보고 에러가 발생하는지 답하라.

만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

#### Questions 2-2)

main.cc 에서 var2->func2() 을 실행 시킬때, Edward 의 func2()만 호출 되게 하려면,

코드를 어떻게 변경시켜야 하는지 설명하시오. 아래는 원하는 콘솔 출력이다.

```
➤ /mnt/d/FINAL/Problem3 ➤ master !3 ?2 ➤ ./bin/main.out
E func2
```

Ans)

#### Questions 3-1)

main.cc 에서 var5->func4() 을 실행 시켜보고 에러가 발생하는지 답하라.

만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

#### Questions 3-2)

main.cc 에서 var5->func4() 을 실행 시킬때, Richard 의 func3() 도 같이 호출 되게 하려면,

코드를 어떻게 변경시켜야 하는지 설명 하시오. 아래는 원하는 콘솔 출력이다.

```
➤ /mnt/d/FINAL/Problem3 ➤ master !3 ?2 ➤ ./bin/main.out
D func4
R func3
```

Ans)

**Questions 4-1)**

main.cc 에서 var4->func1() 을 실행 시켜보고 에러가 발생하는지 답하라.

만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

**Questions 4-2)**

왜 James 의 func1()이 아닌 Richard 의 func1() 만 호출되는지 설명하여라.

---

**Ans)**

---

**Questions 5)**

main.cc 에서 ((James\*) var3)->func4() 을 실행 시켜보고 에러가 발생하는지 답하라.

만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

---

**Ans)**

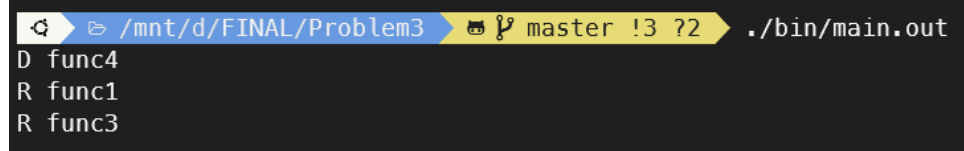
---

**Questions 6-1)**

main.cc 에서 ((Donard\*) var3)->func4() 을 실행 시켜보고 에러가 발생하는지 답하라.  
만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라

**Questions 6-2)**

main.cc 에서 ((Donard\*) var3)->func4() 을 실행 시킬 때, Richard 의 모든 멤버 함수들이 연달아 호출 되도록 하려면, 코드를 어떻게 변경시켜야 하는지 설명 하시오. 아래는 원하는 콘솔 출력이다.



```
➤ /mnt/d/FINAL/Problem3 ➤ master !3 ?2 ➤ ./bin/main.out
D func4
R func1
R func3
```

---

**Ans)**

---

**Questions 7)**

main.cc 에서 ((Edward\*) var3)->func4() 을 실행 시켜보고 에러가 발생하는지 답하라.  
만약 에러가 발생한다면 왜 에러가 발생하는지 서술하여라 (Hint: Down-Casting)

---

**Ans)**

---