**2016, 가을학기, 고급 객체지향 프로그래밍 시험 1**

1. 두 배열 내 원소끼리의 크기를 비교하는 함수를 만들어라. (20점)
   1. 크기가 같은 Integer 배열 A, B 가 있다고 할 때, 함수 pairwiseComparison는 A, B 를 파라미터로 받아, 같은 index를 가지는 원소끼리 크기를 비교하고, 비교 결과를 또 다른 배열로 만들어 주는 함수이다.
   2. 결과 배열 “C”는 다음과 같은 규칙으로 만든다.   
      C[i]=1 if A[i]< B[i]  
      C[i]=0 if A[i]= B[i]  
      C[i]=-1 if A[i]> B[i]
   3. 예를 들어, A={2,5,6}, B={4,2,6} 이라면 결과 C={1,-1,0} 이다.
   4. 위 기능을 다음 두 가지 형태로 구현하라.   
      int\* pairwiseComparison1(int\* A, int\* B, int size) //결과 배열 C를 리턴함. Dangling pointer 주의!!  
      void pairwiseComparison2(int A[], int B[], int\* C, int size) //결과 배열 C가 파라미터로 넘어감
   5. Main 함수에서는 사용자 입력으로 배열의 크기를 받고 이 크기만큼의 배열 A, B 를 “만든”다음, 1부터 100까지의 임의의 숫자로 이 배열을 채운다.
   6. 위 두 함수를 호출하여 결과 배열 C를 구성한 다음 다음과 같이 화면에 출력하여 Test한다.

**Size of the array? 3**

**[6 7 2] and [1 2 9] are created.**

**The comparison result of the first function is [-1 -1 1].**

**The comparison result of the second function is [-1 -1 1].**

1. 행렬 2×2 행렬 A 에 대해 A의 행렬식 (Determinant) 는 det(A)로 나타내고 다음과 같이 계산할 수 있다. (20점)

2차원배열을 입력하시오.

(1,1): 4

(1,2): 7  
(2,1): 2  
(2,2): 6  
원본 행렬:  
4 7  
2 6  
역행렬:  
0.6 -0.7  
-0.2 0.4

혹은

(1,1): 3  
(1,2): 4  
(2,1): 6  
(2,2): 8  
원본행렬:   
3 4  
6 8  
에러: 역행렬이 존재하지 않음



그리고 A의 역행렬은 (inverse of A)은 A-1로 나타내고 다음과 같이 계산할 수 있다.

\mathbf{A}^{-1} = \begin{bmatrix}
a & b \\ c & d \\ 
\end{bmatrix}^{-1} =
\frac{1}{\det(\mathbf{A})} \begin{bmatrix}
\,\,\,d & \!\!-b \\ -c & \,a \\ 
\end{bmatrix} =
\frac{1}{ad - bc} \begin{bmatrix}
\,\,\,d & \!\!-b \\ -c & \,a \\ 
\end{bmatrix}. det(A)=0이면 역 행렬 존재하지 않음

* 1. 2×2 2차원 행렬을 입력 파라미터로 받아서 이 행렬의 행렬식을 반환해 주는 함수, matrixDet를 만들어라.
  2. 그리고 이 함수를 이용해 이 행렬이 역함수가 존재하는지 아닌지를 검사하는 함수 isInversable 함수를 만들어라 (행렬식의 값이 0이면 역행렬이 존재하지 않는다). 이 함수는 역행렬이 존재하면 true를 아니면 false를 반환한다.
  3. 위 두 함수를 이용해 행렬의 역행렬을 파라미터로 반환해 주는 함수 matrixInverse를 만들어라. 함수는 파라미터 2개를 가지는데, 첫 번째 파라미터는 입력 행렬이고 (2×2행렬), 두 번째 파라미터는 구해진 역행렬을 저장하고 반환할 목적의 파라미터이다 (2×2 행렬). 이 함수의 리턴 값은 int형이다. 만약 이 행렬이 역행렬이 존재한다면 리턴값은 1, 아니라면 0을 반환하라.
  4. Main 함수에서는 사용자 입력을 받아 2×2행렬을 채우고, matrixInverse 함수만을 호출하여 역행렬을 구하여 옆처럼 결과를 출력하라.

1. 2번의 문제를 객체를 사용하여 구현할 수 있다. (20점)
   1. “행렬”을 객체화 한다면, “행렬” 객체는 데이터 (a, b, c, d)와 수행할 수 있는 행동 (matrixDet, isInversable, matrixInverse 함수들 및 생성자)을 가질 것이다.
   2. 2번의 결과를 그대로 얻을 수 있게 Matrix class를 구성하고 main에서 test하라.
   3. 단, 행렬의 초기화 시, 숫자 4개를 사용자 입력으로 받은 후 생성자의 인자로 사용해야 함. (생성자 내에서 사용자 입력 받지 말 것)
   4. 데이터 및 함수 Encapsulation을 잘 할 것 (점수에 포함). (hint: 외부에서 직접 호출되지 않는 함수들은 모두 hide)
2. 서점의 책 관리를 위해 다음과 같이 책 클래스를 구현하려고 한다. (20점)
   1. Book 클래스의 멤버 변수
      1. 책 이름을 저장할 수 있는 char array혹은 string 형 변수 혹은 문자열 literal을 할당할 수 있는 char\* 형 변수
      2. 책 가격을 저장하는 unsigned int형 변수
      3. 책 평점을 저장하는 unsigned int형 변수
      4. 현재까지 평점 개수를 저장하는 unsigned int 형 변수
      5. 현재 재고 개수를 저장하는 unsigned int형 변수
   2. Book 클래스의 멤버 함수
      1. 기본 생성자. 책 이름은 “anybook”, 평점 점수 및 개수 0, 가격은 100원, 현재 재고는 1로 초기화
      2. 책 이름만 파라미터로 받는 생성자. 평점 점수 및 개수 0, 가격은 100원, 현재 재고는 1로 초기화
      3. 5가지 정보 모두를 받는 생성자.
      4. 다른 Book 객체를 복사하는 복사 생성자
      5. 이름을 바꿔줄 수 있는 메소드 (changeName)
      6. 평점을 하나 추가해 주는 메소드 (addPoint) 이 함수는 평점 개수를 하나 추가하고 지금까지 평점 평균으로 평점을 갱신해 주어야함
      7. 책 입고 메소드 (addBook) 재고 수 하나 증가
      8. 책 출고 메소드 (releaseBook) 재고 수 하나 감소, 0 미만이면 감소하지 말고 경고메시지 출력
      9. 책 정보를 프린트 해 주는 메소드 (printInfo)
   3. 위 클래스를 구현한 다음 main 함수에서 다음과 같은 일을 수행하라. 예상 결과는 오른쪽과 같음

**int main() {  
 Book book1;  
 Book book2(“Bible”,50000, 75, 2, 5);  
 Book book3(book2);  
 book1.printInfo();**

**book2.printInfo();**

**book3.printInfo();**

**book1.changeName(“Bible 2”);**

**book1.addPoint(50);**

**book1.addPoint(100);**

**book1.addBook();**

**book1.releaseBook();**

**book1.releaseBook();**

**book1.releaseBook();**

**book1.printInfo();**

**return 0;**

**}**

**Anybook 책 생성**

**Bible 책 생성  
Bible 책 생성  
Anybook:평점(0), 가격(100), 재고(1)**

**Bible:평점(75), 가격(50000), 재고(5)**

**Bible:평점(75), 가격(50000), 재고(5)**

**Anybook -> Bible 2로 이름 바뀜**

**Bible 2:평점(50) 추가, 평점평균(50)**

**Bible 2:평점(100) 추가, 평점평균(75)**

**Bible 2:재고 하나 추가**

**Bible 2:재고 하나 감소**

**Bible 2:재고 하나 감소**

**Bible 2:재고가 없어 출고 불가!!!**

**Bible 2:평점(75), 가격(100), 재고(0)**

\* 클래스 구현 시 멤버 변수와 함수들을 private, public으로 적절히 구분해 구현해야 함. 그리고 책 이름 복사 및 할당 시 자기가 원하는 string 관리 방법 사용.

1. 위의 Book 을 관리할 수 있는 StoreManager 클래스를 구현하라. (20점)
   1. 기본적으로, 매장에 가지고 있는 책의 리스트를 관리해 줌.
   2. 다음 코드가 실행되게 클래스를 적절히 구현하고, Main에서 같은 코드로 test하라.  
      \*리스트를 배열로 구현하면 기본점수. 책의 개수에 따라 크기가 변하게 구현하면 추가점수.

**StoreManager KHB;**

**Book book1(“Peter Pan”, 3000, 0,0,1);**

**KHB.stockUpBook(book1); //피터팬 책 입고**

**KHB.stockUpBook(Book(“Bible”,50000, 75, 2, 5)); //Anonymous Object로 책 입고시킴**

**//아래 두 줄 실행되면 보너스 점수**

**Book batchBook[2] = {Book(“Don Quixote”, 3000,90,4,2), Book(“Moby Dick”, 200, 45,2,3)}; KHB.stockUpBook(batchBook, 2); //위 책 2개 한꺼번에 입력**

**KHB.printStock(); //입력한 책 4가지 모두 출력 (Book의 printInfo 함수 사용)**