**2014, 가을학기, 고급 객체지향 프로그래밍 시험 1**

1. 다음 두 함수를 구현하라. (30점)
   1. 두 3차원 벡터, **a** = (*a*1, *a*2, *a*3)와 **b** = (*b*1, *b*2, *b*3) 에 대해서 두 벡터의 합 **a+b**는 산술적으로 다음과 같이 구한다.

합

[6 7 2] + [1 2 9] = [7 9 11]

역행렬

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

**a+b**=(*a*1+b1*, a2+b2, a3+b3)*

두 벡터가 주어졌을 때 이들의 합을 구하는 함수를 구현하라. 이 함수의 prototype는 다음과 같이 구성하여야 한다.

**int \* matrixSum(int\* a, int\* b);**

여기서 입력 파라미터 a, b는 두 3차원 벡터로 크기가 3인 1차행렬이 된다. 이 함수는 내부적으로 새로운 벡터 하나(사이즈가 3인 1차행렬)를 생성하고 여기에 합의 결과를 저장한 다음 이 행렬의 첫 번째 element를 가리키는 포인터를 리턴하여야 한다.   
힌트) 함수 안에서 dynamic memory allocation 사용

* 1. 행렬 2×2 행렬 A 에 대해 A의 역행렬은 (inverse of A)은 A-1로 나타내고 다음과 같이 계산할 수 있다.

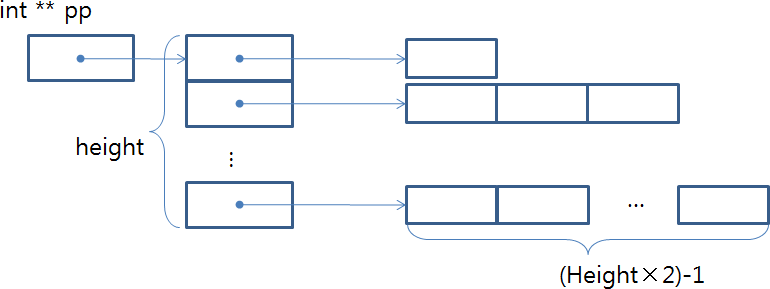
\mathbf{A}^{-1} = \begin{bmatrix}
a & b \\ c & d \\ 
\end{bmatrix}^{-1} =
\frac{1}{\det(\mathbf{A})} \begin{bmatrix}
\,\,\,d & \!\!-b \\ -c & \,a \\ 
\end{bmatrix} =
\frac{1}{ad - bc} \begin{bmatrix}
\,\,\,d & \!\!-b \\ -c & \,a \\ 
\end{bmatrix}. det(A)=0이면 역 행렬 존재하지 않음

2×2 2차원 행렬을 입력 파라미터로 받아서 이 행렬의 역행렬을 파라미터로 반환해 주는 함수, matrixInverse를 만들어라. 함수는 파라미터 2개를 가지는데, 첫 번째 파라미터는 입력 행렬이고 (2×2행렬), 두 번째 파라미터는 구해진 전치행렬을 저장하고 반환할 목적의 파라미터이다 (2×2 행렬). 이 함수의 리턴 값은 int형이다. 만약 이 함수가 역함수가 존재한다면 리턴값은 1, 아니라면 0을 반환하라.

위 두 함수를 만든 다음 옆과 같이 출력되게 적절히 main함수를 구성하라. A번의 두 입력 벡터는 1~10 사이의 임의의 정수로 구성하고 B번의 입력 행렬은 사용자 입력을 받아 채워 넣어라.

1. Trinomial triangle을 double pointer를 이용한 불규칙 배열로 구성하라. 높이가 5인 Trinomial triangle은 다음과 같다. (20점)

\begin{matrix}
 & &  &  & 1\\
 & &  & 1& 1&1\\
 & & 1& 2& 3&2&1\\
 &1& 3& 6& 7&6&3&1\\
1&4&10&16&19&16&10&4&1\end{matrix}

Trinomial triangle의 각 원소는 [바로 위의 원소], [왼쪽 위 원소], [오른쪽 위 원소]의 합으로 나타낸다. 셋 중 하나가 없다면 없는 것은 0으로 가정한다. 가장 위 level의 첫 번째 원소는 1로 시작한다. 프로그램은 우선 사용자로부터 triangle의 높이를 입력 받는다. 그런 다음, dynamic memory allocation을 사용하여 옆 그림과 같은 불규칙한 배열 구조를 만든다.

배열 구조를 만들 때 인덱싱을 위한 int형 변수 외에는 어떠한 다른 변수도 새롭게 만들지 않아야 하고 pp와 포인터연산과 간접연산자(\*) 만을 사용해서 모든 부분을 구현해야 한다. 배열 구조를 만든 후 배열 안의 내용물을 채워 넣는 구문을 구현한다. 이 때도 마찬가지로 pp와 포인터연산과 간접연산자(\*) 만을 사용 모든 것을 구현한다. 배열구조를 만든 후 위 배열을 적절한 구조로 출력한다. 출력 시는 pp와 배열 첨자연산자([])를 사용하여야 한다. 옆은 출력 결과의 예이다.

**Height? 3**

**1**

**1 1 1**

**1 2 3 2 1**

1. 서점의 책 관리를 위해 다음과 같이 책 클래스를 구현하려고 한다. (30점)
   1. Book 클래스의 멤버 변수
      1. 책 이름을 저장할 수 있는 char array혹은 string 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 (addPoint) 이 함수는 평점 개수를 하나 추가하고 지금까지 평점 평균으로 평점을 갱신해 줘아함
      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**

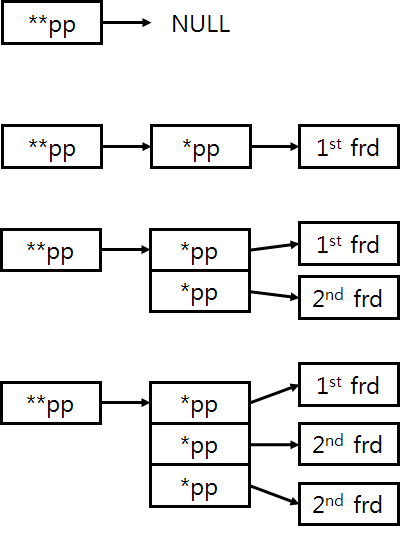
**Bible 2:재고 하나 감소**

**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. 서점의 사용자 클래스를 만들고 각 사용자가 구입한 책을 3에서 만든 Book 클래스를 이용하여 관리하라. (20점)
   1. User 클래스는 다음과 같은 멤버 변수를 가짐
      1. 사용자 이름, char array혹은 string 형
      2. 구입한 책 개수
      3. 구입한 책 리스트: 이 사용자가 구입한 책들의 Book 오브젝트 리스트. 기본적으로 구입한 Book 들을 가리키는 포인터들을 모아놓은 배열. 리스트의 크기가 가변적이며, 책이 추가되면 ‘책 수 \* 1’ 크기의 포인터 배열로 구성됨.
   2. 멤버 함수
      1. 기본 생성자에서 이름을 “Anonymous”, 리스트를 NULL 로 초기화
      2. 이름 변경 메소드 (changeName)
      3. 구입한 책 추가 메소드 (addBook): 파라미터로 Book 객체를 하나 받아서 자신의 구입 리스트 갱신 (파라미터로 받은 오브젝트를 가리키는 포인터를 추가). 구입 책 개수도 갱신. 파라미터로 받은 책 오브젝트의 출고 함수도 호출해야 함. 만약에 해당 책의 재고가 없다면 0을 리턴, 아니면 1을 리턴. 책이 추가되면 현재 책 수 + 1 크기만큼의 새로운 리스트를 할당받아 기존의 리스트를 복사하고, 마지막 엘리먼트에 파라미터로 받아온 객체의 주소를 대입함 (오른쪽 그림 참조). 추가하려는 오브젝트가 이미 있는지 검사는 하지 않아도 됨.

3rd frd

* + 1. 구입 책 목록을 출력해 주는 함수.
  1. 위를 구현 한 다음 main 함수에서 다음과 같은 일을 수행하라. 예상결과는 오른쪽과 같음

**int main() {**

**Book book1(“Peter Pan”, 3000, 0,0,1) ;  
 Book book2(“Bible”,50000, 75, 2, 5);**

**User user1;  
 user1.changeName(“Jane”);**

**user1.addBook(book2);**

**user1.addBook(book1);**

**user1.addBook(book1);**

**user1.printBooks();**

**return 0;**

**}**

**Peter Pan 책 생성**

**Bible 책 생성**

**Anonymous 사용자 생성**

**Anonymous -> Jane 이름 바뀜  
Bible:재고 하나 감소**

**Jane:Bible 책 구입**

**Peter Pan:재고 하나 감소**

**Jane:Peter Pan 책 구입**

**Peter Pan: 재고가 없어 출고 불가!!!**

**Jane:Peter Pan 책 구입 실패 (재고없음)**

**Jane:구입목록**

**Bible:평점(75), 가격(50000), 재고(4)**

**Peter Pan: 평점(0), 가격(3000), 재고(0)**