**2016, 봄 학기, 고급 객체지향 프로그래밍 시험 1**

1. **Bell triangle을 double pointer를 이용한 불규칙 배열로 구성하라. 높이가 7인 Bell Triangle은 다음과 같다. (25점)**

1

1 2

2 3 5

5 7 10 15

15 20 27 37 52

52 67 87 114 151 203

203 255 322 409 523 674 877

Bell triangle의 각 원소의 값은 왼쪽 원소의 값과 왼쪽 위 원소의 값의 합과 같다. 단, 각 행의 첫 번째 원소의 값은 윗 행의 마지막 열에 있는 원소의 값과 같다. 또한 가장 위의 원소 값은 1로 시작한다. 프로그램은 먼저 사용자로부터 높이를 입력받고, 입력 받은 높이에 맞게 불규칙한 배열 구조를 동적 메모리 할당으로 만든다.

**Height: 5**

**1**

**1 2**

**2 3 5**

**5 7 10 15**

**15 20 27 37 52**

배열의 구조를 만들 때 인덱싱을 위한 int형 변수 외에는 어떠한 다른 변수도 새롭게 만들지 않아야 하고, pp와 포인터연산과 간접 연산자(\*) 만을 사용해서 모든 부분을 구현해야 한다. 배열 구조를 만든 후, 배열 안의 내용물을 채워 넣는 구문을 구현하며, 이 때도 마찬가지로 pp와 포인터 연산과 간접연산자(\*) 만을 사용하여 구현한다. 배열 구조를 만든 후 위 배열을 적절한 구조로 출력한다. 출력시는 pp와 배열 첨자연산자([ ])를 사용해야 하며 옆은 출력 결과의 예이다.

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. **지인관리 프로그램 KHUConnect를 만들려고 한다. KHUConnect에서 쓰일 친구 정보저장 클래스 KTFriend 클래스를 구현하고 테스트하라. (25점)**
   1. KTFriend 클래스의 멤버 변수

**int main() {  
 KTFriend haha;**

**KTFriend nana(“Nana”);**

**KTFriend kaka(“Kaka”, 4);**

**haha.addFriend();**

**haha.setName(“Haha”);**

**nana.deleteFriend();**

**kaka.setStatus(2);**

**haha.printFriendInfo();**

**nana.printFriendInfo ();**

**kaka.printFriendInfo ();**

**return 0;**

**}**

* + 1. char name: 친구 이름
    2. int numFriend: 이 친구의 인간관계 수
    3. int status: 0이면 offline, 1이면 online, 2이면 busy
  1. 멤버 함수
     1. void addFriend(): 친구추가 함수. 친구수를 1 증가
     2. void deleteFriend(): 친구빼기 함수. 친구수를 1 감소 (친구가 없을 때는 에러 메시지)
     3. void setStatus(int): 입력 파라미터로 status 값 변경
     4. int getStatus(): status 리턴
     5. void setName(char\*): 파라미터로 받은 이름으로 이름 바꾸기
     6. 기본 생성자: 이름을 Anonymous로, 친구를 0으로, status를 online으로 초기화
     7. 생성자 1: 이름을 입력으로 받아 그것으로 이름을 초기화. 친구수는 0, status는 online
     8. 생성자 2: 이름과 친구수와 status모두를 입력 받아 초기화
     9. void printFriendInfo(): 현재 이 친구의 상황을 출력해 주는 함수

**Nana의 친구수가 0이라서 지울 수 없음**

**[Haha] 친구수: 1, 현재 online**

**[Nana] 친구수: 0, 현재 online**

**[Kaka] 친구수: 4, 현재 busy**

* 1. 위 클래스를 구현한 다음 main 함수에서 다음과 같은 일을 수행하라. 예상 결과는 오른쪽과 같음  
      \* 클래스 구현 시 멤버 변수와 함수들을 private, public으로 적절히 구분해 구현해야 함.

1. **KHUConnect 에서는 쿠-게시판을 만들 수 있다. 이 쿠-게시판을 표현하기 위한 클래스 KCBoard을 구현하고 테스트하라. (25점)**
   1. KCBoard은 다음과 같은 데이터를 변수형태로 관리하여야 한다.
      1. char \*\* postList: 글 목록 (char의 2차원 array 형태로 관리. 각 줄이 글 하나를 의미하고 이 글 들의 집합이 2차원 배열이 됨. Dynamic memory allocation 사용. 문장 시작 지점에 이 문장을 입력한 Friend의 이름도 추가해야 함.)
      2. int numPost: 게시판의 글 수
      3. KTFriend \* participant: 게시판참여 사용자 목록 (3번에서 만든 KTFriend 클래스 사용. KTFriend의 리스트를 배열로 만듬. 최대 사용자 수를 10명으로 하고 배열의 크기는 10으로 함)
   2. 또한, 다음과 같은 일을 함수형태로 수행하여야 한다.
      1. void writePost(KTFriend&, char\*): 사용자가 대화를 입력함. chatList 업데이트 하야 함
      2. void printPostList(): 대화 출력
      3. void printFriends(): 현재 사용자 정보 출력
      4. bool addFriend(KTFriend&): 대화 참여 사용자 추가
      5. bool removeFriend(char\*): 사용자 퇴장
      6. 생성자들
   3. 위 맴버 변수들과 함수들을 적절히 구현하여 다음과 같은 Main 함수의 코드가 동작하여 다음과 같은 결과를 화면에 출력하게 만들어라.

**int main() {  
KCBoard board1;**

**KTFriend haha(“Haha”, 2);**

**KTFriend kaka(“Kaka”, 4);**

**KTFriend kuku(“Kuku”);**

**board1.addUser(haha);**

**board1.addUser(kaka);**

**board1.addUser(kuku);**

**printFriends();**

**writePost(haha,”여기는 C++ 스터디 방임다”);**

**writePost(kaka,”다들 시험공부는 많이 하셨남?”);**

**writePost(kuku,”씨뿔뿔 넘 어려워..흑흑”);**

**writePost(haha,”난 씨뿔뿔 넘 쉬워~~메롱”);**

**writePost(haha,”난 술 한잔 하로 감. 같이 갈사람”);**

**if(removeFriend(“Haka”) == false) {**

**cout << “게시판에 없는 사용자를 지우려고 함” << endl;**

**}**

**removeFriend(“Haha”);**

**printPostList();**

**writePost(kaka,”나도 같이가~~~”);**

**removeFriend(“Kaka”);**

**printFriends();**

**printPostList();**

**return 0;**

**}**

**Haha가 대화에 참여했습니다.**

**Kaka가 대화에 참여했습니다.**

**Kuku가 대화에 참여했습니다.**

**----게시판 사용자 정보----**

**[Haha] 친구수: 2, 현재 online**

**[Nana] 친구수: 4, 현재 online**

**[Kaka] 친구수: 0, 현재 online**

**게시판에 참여하지 않은 사용자를 지우려고 함**

**Haha가 퇴장했습니다.**

**----Post List----**

**Haha: 여기는 C++ 스터디 방임다**

**Kaka: 다들 시험공부는 많이 하셨남?**

**Kuku: 씨뿔뿔 넘 어려워..흑흑**

**Haha: 난 씨뿔뿔 넘 쉬워~~메롱**

**Haha: 난 술 한잔 하로 감. 같이 갈사람**

**Kaka가 퇴장했습니다.**

**----게시판 사용자정보----**

**[Nana] 친구수: 4, 현재 online**

**----Post List----**

**Haha: 여기는 C++ 스터디 방임다**

**Kaka: 다들 시험공부는 많이 하셨남?**

**Kuku: 씨뿔뿔 넘 어려워..흑흑**

**Haha: 난 씨뿔뿔 넘 쉬워~~메롱**

**Haha: 난 술 한잔 하로 감. 같이 갈사람**

**Kaka: 나도 같이가~~~**