Name: Jiho Choi **ID:** 1801787.

1. Write delete statements that correctly delete the following dynamically allocated entities. Assume is all in a single function. Hint: draw out the memory layout on scratch paper.

```
int *p1 = new int[10];

int **p2 = new int*[5];
for (int i = 0; i < 5; i++)
      p2[i] = new int;

int *p3[15];
for (int i = 0; i < 15; i++)
      p3[i] = new int[5];

int *p4 = new int;
int *temp = p4;
p4 = p1;
p1 = temp;
```

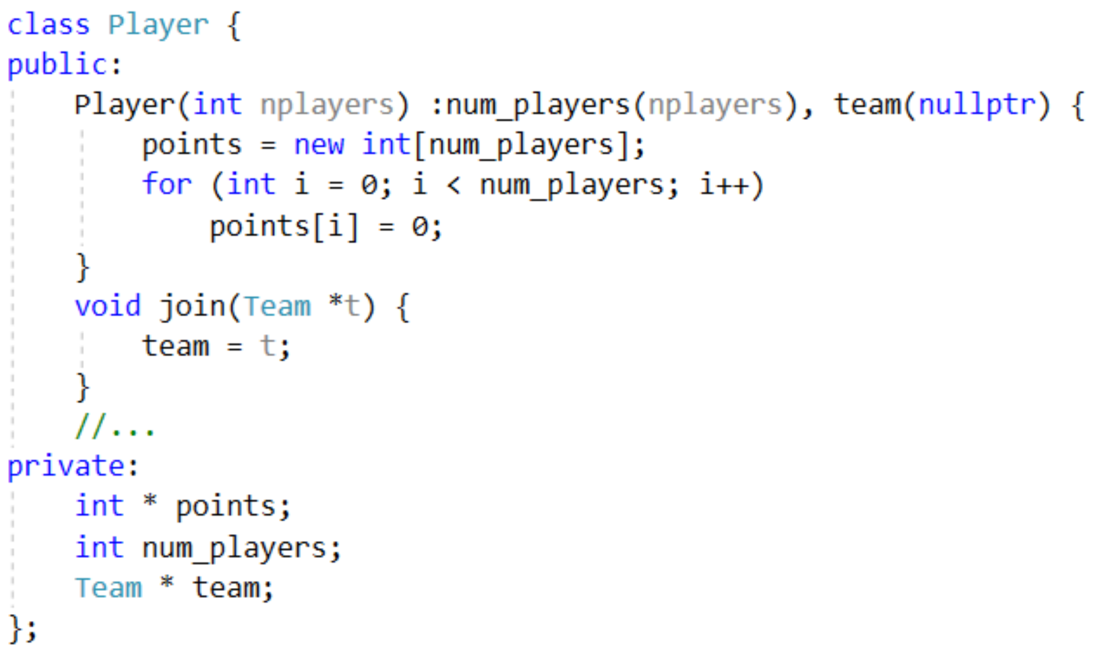
```
delete [] p4;
for (int i=0; i<5; i++){
      delete [] p2[i];
}
    delete [] p2;
for (int i=0; i< 15; i++)
    delete [] p3[i];

delete p1;
```

2. Consider the code fragment below. It is supposed to construct a 3x4 (3 rows 4 columns) 2-level array of integers and set each value to zero. However, as given it does not. Add the proper dereferences (*) or references (&) to make this code work properly:

```
int   **rows;    int   *col1;    int   *col2;    int   *col3;

rows    = new   int  *[3];       // Create 3 pointers to columns
col1    = new   int   [4];       // Create first row with 4 elements
col2    = new   int   [4];       // Create second row with 4 elements
col3    = new   int   [4];       // Create third row with 4 elements

*(  rows + 0 ) = &col1   [0];    // Point to first row
*(  rows + 1 ) = &col2   [0];    // Point to second row
*(  rows + 2 ) = &col3   [0];    // Point to third row

for (int i = 0; i<3; i++)
      for (int j = 0; j<4; j++)
          *(  *(    rows +  i   ) +  j   ) = 0 // same as rows[i][j] = 0;
```

Name: Inho Choi **ID:** 1801787 .

3. Consider the following class definition for a Player that maintains the number of points scored against other players in a game. Players have the ability to change teams. There are always at least 2 players.



```
class Player {
public:
    Player(int nplayers) :num_players(nplayers), team(nullptr) {
        points = new int[num_players];
        for (int i = 0; i < num_players; i++)
            points[i] = 0;
    }
    void join(Team *t) {
        team = t;
    }
    //...
private:
    int * points;
    int num_players;
    Team * team;
};
```

a. Will the following function result in any errors (either syntactical or logical)? If so, what?

```
void foo() {
    Player p;
}
```

This function result will occur error because object p did not have value. It has to deliver any integer. like Player p(10);

If the class player have default constructor, this function will not have error.

Name: Inho Choi **ID:** 1801787.

b. Will the following function result in any errors (either syntactical or logical)? If so, what and how would you address it?

```
void toe() {
    Player p(10);
}
```

this function result will not occur any errors because the object p have the value that is delivered.

c. Assuming any issues from previous parts have been addressed, will the following function result in any errors (either syntactical or logical)? If so, what and how would you address it?

```
void bar() {
    Player p(10);
    Player x = p;
}
```

This function resule will not occur any errors. The object P(10) have the integer value that is deliered constructor and Player x=P; will call default copy constuctor which is shallow copy.

d. Does Player require the definition of Team in order to compile?

Player does not requiro the definition of Team in order to complie.
It is enough to declare "Team" not definition.

Name: Inho Choi **ID:** 1801787.

4. Referring to the Player class in the previous problem:
   a) Implement a copy constructor for the Player . Assume this is done outside the class definition.

```
Player:: player (const Player& copy) {
  num_players = copy.num_players;
   points = new int [num_players];
   for (int i=0; i<num_players; i++) {
         points[i] = copy.points[i];
   }
    team = copy.team;
}
```

   b) Overload the assignment operator for the Player. Assume this is done outside the class definition.

```
Player& player ::operator= (const Player& copy) {
        if (&copy == this) {
         return (*this);
       }
       delete [] copy.points;
      num_players = copy.num_players;
       points = new int[num_players];
    for(int i=0; i<num_players; i++) {
          points[i] = copy.points[i];
    }
           team = copy.team;
return (*this);
}
```

   c) Implement the destructor for Player, Assume this is done outside the class definition.

```
player :: ~player() {
            delete [] points;


}
```