Lab 05 (Dated 16-10-2020)

1. Class Player

Create class Player. Player object has data members: player number (const), number Of matches, dynamic array to store scores and dynamic character array to store status of each match {'O' out, 'N' notout, 'D' do not bat}. Player class has static member number of players. Create member functions for:

1. Constructor with one integer parameter number of matches, create dynamic arrays accordingly
2. Read scores and status, according to number of matches
3. Private member function to count number of innings. For number of innings count all matches with status 'O'
4. Calculate player average, divide total score by number of innings. Call private member function to count number of innings
5. Show function to display players information according to the format
6. Static function to return no of players

Write main function. Create 3 dynamic objects of Player by reading 3 positive numbers, representing number of matches played by each player. Next read "n" number of operations. Next n lines contain operation number and player number (if required) . For each operation perform specific operation:

1. For operation number 1, calculate player (according to player number) average in integer and print
2. For operation number 2, call show function according to player number
3. Show total no of players. For operation 3, single input required
4. Show average of specific player
5. Delete player
6. Again create player by reading number of matches (three inputs, operation number, player number and number of matches)
7. Read scores and status

Input Format

```
2 4 3
7
3
6 1
3 50 O 40 O
6 2
3 50 O 40 O 60 N
2 2
2
4 2
5 3
3
```

Output Format

```
Number of Players: 3
Player 2 has played 3 matches and 2 innings
Scores: 50 40 60
Average: 75
Number of Players: 2
```

Solution:

```
class Player{
    const int playerNo;
    int noOfMatches;
    int *scores;
    char *status;
    static int noOfPlayers;
```

```
    int countInnings() const{
        int count = 0;
        for (int i=0;i<noOfMatches;i++)
            if (status[i]=='O')
                count++;
        return count;
    }
    int getTotalScores() const{
        int total = 0;
        for (int i=0;i<noOfMatches;i++)
            total += scores[i];
        return total;
    }
public:
    Player(int pNo, int noOfMatches): playerNo(pNo){
        this->noOfMatches = noOfMatches;
        scores = new int [noOfMatches];
        status = new char [noOfMatches];
        noOfPlayers++;
    }
    void read(){
        for (int i=0;i<noOfMatches;i++)
            cin >> scores[i] >> status [i];
    }
    int calculateAverage() const{
        return getTotalScores () / countInnings();
    }
   void show() const{
       cout << "Player " << playerNo << " has played " << noOfMatches <<
           " matches and " << countInnings() << " innings\nScores: ";
       for (int i=0;i<noOfMatches;i++)
           cout << scores[i] << ' ';
       cout << '\n';
   }
    ~Player(){
        noOfPlayers--;
    }
   static int getNoOfPlayers(){
       return noOfPlayers;
   }
};
int Player:: noOfPlayers = 0;
void performOperations(Player *p1, Player *p2, Player *p3){
    int operationNo, playerNo, noOfMatches;
    cin >> operationNo;
    if (operationNo==1){
        cin >> playerNo;
        if (playerNo==1)            cout << p1->calculateAverage();
        else if (playerNo==2)       cout << p2->calculateAverage();
        else                        cout << p3->calculateAverage();
    }
    else if (operationNo==2){
```

```
        cin>>playerNo;
        if (playerNo==1)            p1->show();
        else if (playerNo==2)       p2->show();
        else                        p3->show();
    }
    else if (operationNo==3)
        cout << "Number of Players: " << Player::getNoOfPlayers() << '\n';
    else if (operationNo==4){
        cin >> playerNo;
        if (playerNo==1)
           cout << "Average: " << p1->calculateAverage() << '\n';
        else if (playerNo==2)
           cout << "Average: " << p2->calculateAverage() << '\n';
        else
           cout << "Average: " << p3->calculateAverage() << '\n';
    }
    else if (operationNo==5){
        cin >> playerNo;
        if (playerNo==1)            delete p1;
        else if (playerNo==2)       delete p2;
        else                        delete p3;
    }
    else if (operationNo==6){
        cin >> playerNo >> noOfMatches;
        if (playerNo==1)            p1 = new Player(1, noOfMatches);
        else if (playerNo==2)       p2 = new Player(2, noOfMatches);
        else                        p3 = new Player(3, noOfMatches);
    }
    else if (operationNo==7){
        cin >> playerNo;
        if (playerNo==1)            p1->read();
        else if (playerNo==2)       p2->read();
        else                        p3->read();
    }
}
int main() {
    int n1, n2, n3, n;
    cin >> n1 >> n2 >> n3;
    Player *p1 = new Player(1, n1);
    Player *p2 = new Player(2, n2);
    Player *p3 = new Player(3, n3);
    cin >> n;
    for (int i=0;i<n;i++)
        performOperations(p1,p2, p3);
    return 0;
}
```

2. Classes and Objects

A class defines a blueprint for an object. We use the same syntax to declare objects of a class as we use to declare variables of other basic types. For example:

```
Box box1;          // Declares variable box1 of type Box
Box box2;          // Declare variable box2 of type Box
```

Kristen is a contender for valedictorian of her high school. She wants to know how many students (if any) have scored higher than her in the 5 exams given during this semester.

Create a class named *Student* with the following specifications:

- An instance variable named *scores* to hold a student's 5 exam scores
- A void input() function that reads 5 integers and saves them to scores
- An int calculateTotalScore() function that returns the sum of the student's scores

Input Format

Most of the input is handled for you by the locked code in the editor.

In the void `Student::input()` function, you must read 5 scores from stdin and save them to your scores instance variable

Constraints

$1 \leq n \leq 100$

$0 \leq examscore \leq 50$

Output Format

In the int Student::calculateTotalScore() function, you must return the student's total grade (the sum of the values in scores)

The locked code in the editor will determine how many scores are larger than Kristen's and print that number to the console

Solution:

```
class Student{
    int scores[5];
    public:
    void input(){
        int x;
        for(int i=0;i<5;i++)
        {
            cin>>x;
            scores[i]=x;
        }
    }
  int calculateTotalScore(){
      int sum=0;
      for(int i=0;i<5;i++)
          sum=sum+scores[i];
      return sum;

    }
};
```