CSE Assignment (C++)

Problem 1

Question:

> *Create a class called Employee that contains the Name and Employee number (type long). Include a member function called getData() to get data from the user for insertion into the object, another function called putData() to display the data. The name should have no embedded blanks. Write a main() program to exercise this class. It should create array of type Employee and then invite the user to input data for up to 10 employees. Finally, it should print out the data for all the employees. (Use setw, setiosflags to format the output)*

Code:

```
#include <iostream>
#include <iomanip>

using namespace std;

class Employee
{
private:
    string Name;
    long Emp_No;

public:
    static short unsigned int put_calls;

    void getData(string, long);
    void putData();
};

void Employee::getData(string peru, long Reg)
{
    Name = peru;
    Emp_No = Reg;
}

void Employee::putData()
{
    cout << setw(2) << setfill('0') << setiosflags(ios::right) << put_calls + 1 << "
Name: " << Name << "\tEmp.No: " << Emp_No;
    cout << endl;

    put_calls += 1;
}

short unsigned int Employee::put_calls = 0;
```

```
int main(int argc, char const *argv[])
{
    int n;

    cout << setiosflags(ios::left) << "How many employee details would you like to
enter? [Max 10] : " << endl;
    cin >> n;

    if (n > 0 && n < 11)
    {

        Employee emparror[n];

        for (int i = 0; i < n; i++)
        {
            string Name;
            long Reg;

            cout << "Enter name for Employee " << i + 1 << " : ";
            cin >> Name;

            cout << "Enter the number for Employee " << i + 1 << " : ";
            cin >> Reg;

            emparror[i].getData(Name, Reg);
        }

        cout << endl
             << endl
             << "--- DETAILS ---" << endl;

        for (int i = 0; i < n; i++)
        {
            emparror[i].putData();
        }
    }

    return 0;
}
```

Output:

```
How many employee details would you like to enter? [Max 10] :
2
Enter name for Employee 1 : BC
Enter the number for Employee 1 : 1
Enter name for Employee 2 : JS
Enter the number for Employee 2 : 2


--- DETAILS ---
```

```
01 Name: BC Emp.No: 1
02 Name: JS Emp.No: 2
```

Problem 2

Question:

> *Write a program that calculates the average of up to 10 English distances input by the user. Create an array of objects of the Distance class with the data members as feets and inches to calculate the average create a member function called avgDistance(). (Hint: 12 inches = 1 Feet)*

Code:

```
#include <iostream>

using namespace std;

class Distance
{
private:
    static unsigned int obj_count;
    double feet;
    double inches;

public:
    Distance()
    {
        feet = 0;
        inches = 0;
        obj_count++;
    }

    Distance(double &Feet, double &Inches)
    {
        feet = Feet;
        inches = Inches;
    }

    void setParams(double &Feet, double &Inches)
    {
        feet = Feet;
        inches = Inches;
    }
    static Distance avgDistance(const Distance objs[]);
    double getFeet()
    {
        return feet;
    }

    double getInches()
```

```
    {
        return inches;
    }
};

Distance Distance::avgDistance(const Distance objs[])
{
    double feet = 0, inches = 0;

    for (int i = 0; i < obj_count; i++)
    {
        inches += objs[i].inches;
        feet += objs[i].feet;

        if (inches > 11)
        {
            feet += (int)inches / 12;
            inches -= ((int)inches / 12) * 12;
        }
    }

    return Distance(feet, inches);
}

std::ostream &operator<<(std::ostream &os, Distance &obj)
{
    return os << obj.getFeet() << " Feet " << obj.getInches() << " Inches ";
}

unsigned int Distance::obj_count = 0;

int main(int argc, char const *argv[])
{
    int n;

    cout << "Enter the number of English distances you would like to enter : ";
    cin >> n;

    double hmm = 1;

    Distance disarray[n], avg(hmm, hmm);

    for (int i = 0; i < n; i++)
    {
        double feet, inches;

        cout << i << endl;
        cin >> feet;
        cin >> inches;

        disarray[i].setParams(feet, inches);
    }
```

```
    avg = Distance::avgDistance(disarray);

    cout << "The average distance is : " << avg;

    return 0;
}
```

Output:

```
Enter the number of English distances you would like to enter : 3
0
12
12
1
3
4
2
4
4
The average distance is : 20 Feet 8 Inches
```

Problem 3

Question:

> *Write a program to create a class HEIGHT with feet and inches as members. Write member functions to READ, find the maximum height (MAX), find the mimimum height (MIN) and average height (AVG) for arrays of object*

Code:

```
#include <iostream>

using namespace std;

class HEIGHT
{
private:
    static unsigned int obj_count;
    double feet;
    double inches;

public:
    HEIGHT()
    {
        feet = 0;
        inches = 0;
        obj_count++;
    }
```

```
    HEIGHT(double &Feet, double &Inches)
    {
        feet = Feet;
        inches = Inches;
    }

    void setParams(double &Feet, double &Inches)
    {
        feet = Feet;
        inches = Inches;
    }
    static HEIGHT avgHeight(const HEIGHT objs[]);
    static HEIGHT minHeight(const HEIGHT objs[]);
    static HEIGHT maxHeight(const HEIGHT objs[]);
    double getFeet()
    {
        return feet;
    }

    double getInches()
    {
        return inches;
    }
};

HEIGHT HEIGHT::maxHeight(const HEIGHT objs[])
{
    double max_ft = 0, max_in = 0;

    for (int i = 0; i < obj_count; i++)
    {
        double feet = objs[i].feet, inches = objs[i].inches;

        if (inches > 11)
        {
            feet += (int)inches / 12;
            inches -= ((int)inches / 12) * 12;
        }

        if (feet >= max_ft && inches >= max_in)
        {
            max_ft = feet;
            max_in = inches;
        }
    }

    return HEIGHT(max_ft, max_in);
}

HEIGHT HEIGHT::minHeight(const HEIGHT objs[])
{
    double min_ft = 9999999, min_in = 9999999;
```

```
    for (int i = 0; i < obj_count; i++)
    {
        double feet = objs[i].feet, inches = objs[i].inches;

        if (inches > 11)
        {
            feet += (int)inches / 12;
            inches -= ((int)inches / 12) * 12;
        }

        if (feet <= min_ft && inches <= min_in)
        {
            min_ft = feet;
            min_in = inches;
        }
    }

    return HEIGHT(min_ft, min_in);
}

HEIGHT HEIGHT::avgHeight(const HEIGHT objs[])
{
    double feet = 0, inches = 0;

    for (int i = 0; i < obj_count; i++)
    {
        inches += objs[i].inches;
        feet += objs[i].feet;

        if (inches > 11)
        {
            feet += (int)inches / 12;
            inches -= ((int)inches / 12) * 12;
        }
    }

    return HEIGHT(feet, inches);
}

std::ostream &operator<<(std::ostream &os, HEIGHT &obj)
{
    return os << obj.getFeet() << " Feet " << obj.getInches() << " Inches ";
}

unsigned int HEIGHT::obj_count = 0;

int main(int argc, char const *argv[])
{
    int n;

    cout << "Enter the number of heights you would like to enter : ";
```

```
    cin >> n;

    double hmmm = 1;

    HEIGHT disarray[n], res(hmmm, hmmm);

    for (int i = 0; i < n; i++)
    {
        double feet, inches;

        cout << i << endl;
        cin >> feet;
        cin >> inches;

        disarray[i].setParams(feet, inches);
    }

    res = HEIGHT::avgHeight(disarray);

    cout << "The average height is : " << res << endl;

    res = HEIGHT::minHeight(disarray);

    cout << "The min height is : " << res << endl;

    res = HEIGHT::maxHeight(disarray);

    cout << "The max height is : " << res << endl;

    return 0;
}
```

Output:

```
Enter the number of heights you would like to enter : 3
0
12
12
1
12
24
2
12
36
The average height is : 42 Feet 0 Inches
The min height is : 13 Feet 0 Inches
The max height is : 15 Feet 0 Inches
```

Problem 4

Question:

> *Write a program to create a class Date with day, month and year as members. Write a member function to validate the dates and display the invalid dates. Test the class with an array of Date objects*

Code:

```
#include <iostream>

using namespace std;

class Date
{
private:
    short int day;
    short int month;
    short int year;

public:
    void setParams(int d, int m, int y);
    int getParam(char p);
    static bool validateDate(Date &obj);
};

void Date::setParams(int d, int m, int y)
{
    day = d;
    month = m;
    year = y;
}

int Date::getParam(char p)
{
    if (p == 'd')
    {
        return day;
    }

    if (p == 'm')
    {
        return month;
    }

    if (p == 'y')
    {
        return year;
    }

    return 0;
}
```

```
bool Date::validateDate(Date &obj)
{
    if (obj.getParam('y') >= 1000 && obj.getParam('y') <= 3000)
    {
        if (obj.getParam('m') >= 1 && obj.getParam('m') <= 12)
        {
            if (obj.getParam('m') == 7)
            {
                if (obj.getParam('d') > 0 && obj.getParam('d') < 32)\
                {
                    return true;
                }
            }
            else if (obj.getParam('m') == 2)
            {
                if (((obj.getParam('y') % 4 == 0) && (obj.getParam('y') % 100 != 0))
|| (obj.getParam('y') % 400 == 0))
                {
                    if (obj.getParam('d') > 0 && obj.getParam('d') < 30)
                    {
                        return true;
                    }
                }
                else
                {
                    if (obj.getParam('d') > 0 && obj.getParam('d') < 29)
                    {
                        return true;
                    }
                }
            }
            else if (obj.getParam('m') < 7 && obj.getParam('m') != 2)
            {
                if (obj.getParam('m') % 2 == 0)
                {
                    if (obj.getParam('d') > 0 && obj.getParam('d') < 31)
                    {
                        return true;
                    }
                }
                else
                {
                    if (obj.getParam('d') > 0 && obj.getParam('d') < 32)
                    {
                        return true;
                    }
                }
            }
            else if (obj.getParam('m') > 7)
            {
                if (obj.getParam('m') % 2 == 0)
                {
```

```
                    if (obj.getParam('d') > 0 && obj.getParam('d') < 32)
                    {
                        return true;
                    }
                }
                else
                {
                    if (obj.getParam('d') > 0 && obj.getParam('d') < 31)
                    {
                        return true;
                    }
                }
            }
        }
    }

    return false;
}

int main(int argc, char const *argv[])
{
    int n;

    cout << "How many dates would you like to test? : ";
    cin >> n;

    Date datarr[n];

    for (int i = 0; i < n; i++)
    {
        int d, m, y;

        cout << "Enter the Day, Date and Year separated by a space -> Count : " << i +
1 << " => " << endl;
        cin >> d >> m >> y;

        datarr[i].setParams(d, m, y);
    }

    for (int i = 0; i < n; i++)
    {
        bool res;

        res = Date::validateDate(datarr[i]);

        if (res == true)
        {
            cout << "The date is valid" << endl;
        }
        else
        {
            cout << "The date is invalid" << endl;
```

```
        }
    }

    return 0;
}
```

Output:

```
How many dates would you like to test? : 2
Enter the Day, Date and Year separated by a space -> Count : 1 =>
29 2 2000
Enter the Day, Date and Year separated by a space -> Count : 2 =>
31 8 2022
The date is valid
The date is valid
```
