6.1 Template classes_Circle

Source Code

Circle.h

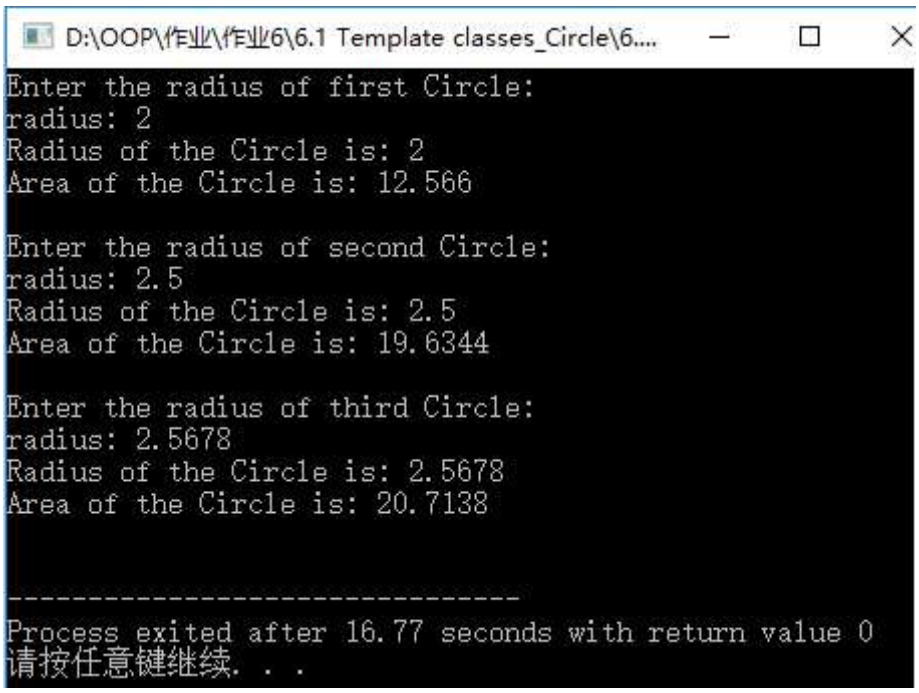
```
#ifndef CIRCLE_H
#define CIRCLE_H

#include<iostream>
using namespace std;

template<typename T>
class Circle
{
public:
        Circle();
        Circle(T radius);
        Circle(const Circle& sec_Circle);

        double Area();
        void input();
        void display();

        T getRadius() const;
        void setRadius(T radius);
private:
        T radius;
};

template<typename T>
Circle<T>::Circle()
{
}

template<typename T>
Circle<T>::Circle(T radius)
{
        this->radius = radius;
}

template<typename T>
Circle<T>::Circle(const Circle& sec_Circle)             // Copy Constructors
{
        this->radius = sec_Circle.getRadius();
}

template<typename T>
double Circle<T>::Area()
{
        double area;
        area =  (3.1415 * radius*radius);
        return area;
}
template<typename T>
void Circle<T>::input()
```

```
{
        cout << "radius: ";
        cin >> this->radius;

        cout << "Radius of the Circle is: " << this->radius << endl;
}
template<typename T>
void Circle<T>::display()
{
        cout << "Area of the Circle is: " << this->Area() << endl;
}

template<typename T>
T Circle<T>::getRadius() const
{
        return radius;
}
template<typename T>
void Circle<T>::setRadius(T radius)
{
        this->radius = radius;
}

#endif
```

main.cpp

```
#include"Circle.h"

int main(int argc, char** argv) {
        Circle<int> intCircle;
        cout << "Enter the radius of first Circle: " << endl;
        intCircle.input();
        intCircle.display();
        cout << endl;

        Circle<float> floatCircle;
        cout << "Enter the radius of second Circle: " << endl;
        floatCircle.input();
        floatCircle.display();
        cout << endl;

        Circle<double> doubleCircle;
        cout << "Enter the radius of third Circle: " << endl;
        doubleCircle.input();
        doubleCircle.display();
        cout << endl;

        return 0;
}
```

Sample Output



6.2

Source Code

Marks.h

```
#ifndef MARKS_H
#define MARKS_H

template<typename T>
class Marks
{
public:
        Marks();        // no-arg constructor
        Marks(T mid, T final, T assignments);
        Marks(const Marks& sec_Marks);          // Copy Constructors

        // Setters and Getters
        T getMid() const;
        void setMid(T mid);
        T getFinal() const;
        void setFinal(T final);
        T getAssignments() const;
        void setAssignments(T assignments);
private:
        T mid;                  //for mid-term exam marks
        T final;                //for final-term exam marks
        T assignments;  //for assignments marks
};

template<typename T>
Marks<T>::Marks()       // no-arg constructor
{
}
template<typename T>
```

```
Marks<T>::Marks(T mid, T final, T assignments)
{
        this->mid = mid;
        this->final = final;
        this->assignments = assignments;
}
template<typename T>
Marks<T>::Marks(const Marks& sec_Marks) // Copy Constructors
{
        this->mid = sec_Marks.getMid();
        this->final = sec_Marks.getFinal();
        this->assignments = sec_Marks.getAssignments();
}

// Setters and Getters
template<typename T>
T Marks<T>::getMid() const
{
        return mid;
}
template<typename T>
void Marks<T>::setMid(T mid)
{
        this->mid = mid;
}
template<typename T>
T Marks<T>::getFinal() const
{
        return final;
}
template<typename T>
void Marks<T>::setFinal(T final)
{
        this->final = final;
}
template<typename T>
T Marks<T>::getAssignments() const
{
        return assignments;
}
template<typename T>
void Marks<T>::setAssignments(T assignments)
{
        this->assignments = assignments;
}

#endif
```

Total.h

```
#ifndef TOTAL_H
#define TOTAL_H

#include<iostream>
#include<string>
#include"Marks.h"
using namespace std;

class Total
```

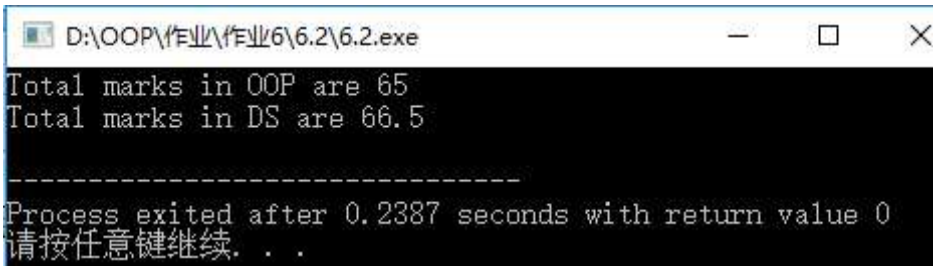
```
{
public:
        Total();                // no-arg constructor
        Total(string course);
        Total(const Total& sec_Total);          // Copy Constructors

        // Setters and Getters
        string getCourse() const;
        void setCourse(string course);

        template<typename T>
        void Total_marks(const Marks<T>& marks);
private:
        string course;  //for course title
};

Total::Total()  // no-arg constructor
{
}
Total::Total(string course)
{
        this->course = course;
}
Total::Total(const Total& sec_Total)    // Copy Constructors
{
        this->course = sec_Total.getCourse();
}

// Setters and Getters
string Total::getCourse() const
{
        return course;
}
void Total::setCourse(string course)
{
        this->course = course;
}

template<typename T>
void Total::Total_marks(const Marks<T>& marks)
{
        T result;
        result = marks.getMid() + marks.getFinal() + marks.getAssignments();
        cout << result;
}

#endif
```

main.cpp

```
#include"Total.h"

int main(int argc, char** argv) {
        Marks<int> marks1(20,35,10);
        Marks<float> marks2(20,36.5,10);

        Total total1("OOP");
        Total total2("DS");
```

```
        cout << "Total marks in " << total1.getCourse() << " are ";
        total1.Total_marks(marks1);
        cout << endl;
        cout << "Total marks in " << total2.getCourse() << " are ";
        total2.Total_marks(marks2);
        cout << endl;

        return 0;
}
```

Sample Output



6.3

Source Code

Gpa.h

```
#ifndef GPA_H
#define GPA_H

#include<iostream>
using namespace std;

const int COURSENUMBER = 5;
const int CREDITHOUR = 3;

template<typename T>
class Gpa
{
public:
        Gpa();          // no-arg constructor
        Gpa(T* result);
        Gpa(const Gpa& sec_Gpa);        // Copy Constructors
        ~Gpa();         // destructor

        // Setters and Getters
        T* getResult() const;
        void setResult(T* result);

        void input();
        void cal_gpa();
private:
        T* result;
};
```

```
template<typename T>
Gpa<T>::Gpa()           // no-arg constructor
{
        this->result = new T[COURSENUMBER];
        for(int i=0;i<COURSENUMBER;i++)
        {
                this->result[i] = 0;
        }
}
template<typename T>
Gpa<T>::Gpa(T* result)
{
        if(result==NULL)
        {
                this->result = new T[COURSENUMBER];
                for(int i=0;i<COURSENUMBER;i++)
                {
                        this->result[i] = 0;
                }
        }
        else
        {
                this->result = new T[COURSENUMBER];
                for(int i=0;i<COURSENUMBER;i++)
                {
                        this->result[i] = result[i];
                }
        }
}
template<typename T>
Gpa<T>::Gpa(const Gpa& sec_Gpa)         // Copy Constructors
{
        delete []result;
        this->result = new T[COURSENUMBER];
        for(int i=0;i<COURSENUMBER;i++)
        {
                this->result[i] = sec_Gpa.getResult()[i];
        }
}
template<typename T>
Gpa<T>::~Gpa()          // destructor
{
        delete []result;
}

// Setters and Getters
template<typename T>
T* Gpa<T>::getResult() const
{
        return result;
}
template<typename T>
void Gpa<T>::setResult(T* result)
{
        delete []result;
        if(result==NULL)
        {
                this->result = new T[COURSENUMBER];
                for(int i=0;i<COURSENUMBER;i++)
                {
                        result[i] = 0;
```

```
                }
        }
        else
        {
                this->result = new T[COURSENUMBER];
                for(int i=0;i<COURSENUMBER;i++)
                {
                        this->result[i] = result[i];
                }
        }
}

template<typename T>
void Gpa<T>::input()
{
        delete []result;
        this->result = new T[COURSENUMBER];
        cout << "Enter student's marks in each course: " << endl;
        for(int i=0;i<COURSENUMBER;i++)
        {
                cin >> result[i];
        }
}
template<typename T>
void Gpa<T>::cal_gpa()
{
        int total_points = 0;
        int total_credits = COURSENUMBER * CREDITHOUR;
        for(int i=0;i<COURSENUMBER;i++)
        {
                if(result[i]=='A' || result[i]>=80)
                {
                        total_points += 12;
                }
                else if(result[i]=='B' || result[i]>=70)
                {
                        total_points += 9;
                }
                else if(result[i]=='C' || result[i]>=60)
                {
                        total_points += 6;
                }
                else if(result[i]=='D' || result[i]>=50)
                {
                        total_points += 3;
                }
                else if(result[i]=='F' || result[i]<=49)
                {
                        total_points += 0;
                }
        }
        double GPA = total_points*1.0 / total_credits;
        cout << "The student's Gpa is " << GPA;
}

#endif
```

main.cpp

```
#include"Gpa.h"

int main(int argc, char** argv) {
        Gpa<int> gpa1;
        gpa1.input();
        gpa1.cal_gpa();
        cout << endl;

        cout << endl;
        Gpa<char> gpa2;
        gpa2.input();
        gpa2.cal_gpa();
        return 0;
}
```

Sample Output