**Name:**

Solution:

#include <iostream>

#include <vector>

using namespace std;

class Student

{

protected: // This needs to be protected so that gradStudent (derived class) can access it

string name;

int age;

int units;

public:

Student()

{

name = "";

age = 0;

units = 0;

}

Student(string n, int a, int u)

{

name = n;

age = a;

units = u;

}

void setName(string n)

{

name = n;

}

void setAge(int a)

{

age = a;

}

void setUnits(int u)

{

if (u < 1 || u > 15)

{

throw out\_of\_range("out\_of\_range");

}

else

{

units = u;

}

}

string getName()

{

return name;

}

int getAge()

{

return age;

}

int getUnits()

{

return units;

}

double tuition(double feePerUnit)

{

return units \* feePerUnit;

}

friend ostream& operator<<(ostream& o, Student& std);

// Q4:

friend bool operator<(Student s1, Student s2);

};

ostream& operator<<(ostream& o, Student& std)

{

o << "Student Name: " << std.name << endl;

o << "Student Age: " << std.age << endl;

return o;

}

bool operator<(Student s1, Student s2)

{

if (s1.units < s2.units)

{

return true;

}

else

{

return false;

}

}

// Q2

class GradStudent : public Student

{

private:

double gradFee;

public:

GradStudent() : Student()

{

gradFee = 0;

}

GradStudent(string n, int a, int u, double g) : Student(n, a, u)

{

gradFee = g;

}

void setGradFee(double g)

{

gradFee = g;

}

double getGradFee()

{

return gradFee;

}

double tuition(double feePerUnit)

{

return Student::tuition(feePerUnit) + gradFee;

}

friend ostream& operator<<(ostream& o, GradStudent& std);

};

ostream& operator<<(ostream& o, GradStudent& std)

{

o << "Student Name: " << std.name << endl;

o << "Student Age: " << std.age << endl;

o << "Student Grad Fee: " << std.gradFee << endl;

return o;

}

template <class T>

void printVector(vector<T> v)

{

for (int i = 0; i < v.size(); i++)

{

cout << v[i] << endl;

}

}

template <class T>

T smallestElement(vector<T> v)

{

T small = v[0];

for (int i = 1; i < v.size(); i++)

{

if (v[i] < small)

{

small = v[i];

}

}

return small;

}

int main()

{

// Q1:

cout << "Q1:" << endl;

Student s1("Khan", 20, 12);

cout << s1;

cout << "Tuition: " << s1.tuition(999) << endl;

cout << endl;

Student s2("Maryam", 21, 11);

cout << s2;

cout << "Tuition: " << s2.tuition(899) << endl << endl;

// s2.setUnits(20); throwing exception

// Q2:

cout << "Q2:" << endl;

GradStudent g1("Hassan", 24, 12, 1000);

cout << g1;

cout << "Tuition: " << g1.tuition(111) << endl;

cout << endl;

GradStudent g2("Sara", 25, 11, 2000);

cout << g2;

cout << "Tuition: " << g2.tuition(123) << endl;

cout << endl;

cout << "Q3:" << endl;

Student\* sPtr;

sPtr = new Student("Ahmed", 20, 14);

cout << "Student Name: " << sPtr->getName() << endl;

cout << "Student Age: " << sPtr->getAge() << endl;

cout << "Student Units: " << sPtr->getUnits() << endl;

cout << "Student Tuition: " << sPtr->tuition(750) << endl;

cout << endl;

sPtr = new GradStudent("Bilal", 25, 10, 1000);

cout << "Student Name: " << sPtr->getName() << endl;

cout << "Student Age: " << sPtr->getAge() << endl;

cout << "Student Units: " << sPtr->getUnits() << endl;

cout << "Student Tuition: " << sPtr->tuition(900) << endl;

// Q4:

cout << endl;

cout << "Q4:" << endl;

if (s1 < s2)

{

cout << s1.getName() << " has less units than " << s2.getName() << endl;

}

else

{

cout << s2.getName() << " has less units than " << s1.getName() << endl;

}

cout << endl;

cout << "Q5:" << endl;

vector<Student> v;

v.push\_back(s1);

v.push\_back(s2);

v.push\_back(g1);

v.push\_back(g2);

printVector(v);

cout << endl;

Student smallest = smallestElement(v);

cout << "Smallest Student: " << smallest.getName() << endl;

}

