CSE 241 PS #4 Solutions

1)

```
#include <iostream>
using namespace std;
const int CLASS_SIZE = 5;
// Problem says this is for a class, rather than one student.
// Strategy: Attack for a single student, then do for an array of N
// students.
//Grading Program
//Policies:
//
//  Two quizzes, 10 points each
//  midterm and final exam, 100 points each
//  Of grade, final counts 50%, midterm 25%, quizes25%
//
//  Letter Grade is assigned:
//  90 or more A
//  80 or more B
//  70 or more C
//  60 or more D
//  less than 60, F
//
// Read a student's scores,
// output record: scores + numeric average + assigned letter grade
//
// Use a struct to contain student record.
struct StudentRecord
{
  int studentNumber;
  double quiz1;
  double quiz2;
```

```
  double midterm;
  double final;
  double average;
  char grade;
};
//prompts for input for one student, sets the
//structure variable members.
void input(StudentRecord& student);
//calculates the numeric average and letter grade.
void computeGrade(StudentRecord& student);
//outputs the student record.
void output(const StudentRecord student);
int main()
{
  StudentRecord student[CLASS_SIZE];
  for(int i = 0; i < CLASS_SIZE; i++)
    input(student[i]);

  // Enclosing block fixes VC++ "for" loop control defined outside loop
  {  for(int i = 0; i < CLASS_SIZE; i++)
    {
      computeGrade(student[i]);
      output(student[i]);
      cout << endl;
    }
  }
  return 0;
}
void input(StudentRecord &student)
{
  cout << "enter the student number: ";
```

```
  cin >> student.studentNumber;
  cout << student.studentNumber << endl;
  cout << "enter two 10 point quizes" << endl;
  cin >> student.quiz1 >> student.quiz2;
  cout << student.quiz1 << " " << student.quiz2 << endl;
  cout << "enter the midterm and final exam grades."
       << "These are 100 point tests\n";
  cin >> student.midterm >> student.final;
cout << student.midterm << " " << student.final
       << endl << endl;
}
void computeGrade(StudentRecord& student)
{
//  Of grade, final counts 50%, midterm 25%, quizes25%
  double quizAvg= (student.quiz1 + student.quiz2)/2.0;
  double quizAvgNormalized = quizAvg * 10;
  student.average = student.final * 0.5 +
                    student.midterm * 0.25 +
                    quizAvgNormalized * 0.25;
  char letterGrade[]= "FFFFFFDCBAA";
  int index = static_cast<int>(student.average/10);
  if(index < 0 || 10 <= index)
  {
    cout << "Bad numeric grade encountered: "
         << student.average << endl
         << "  Aborting.\n";
    abort();
  }
  student.grade = letterGrade[index];
}
void output(const StudentRecord student)
```

```
{
  cout << "The record for student number: "
       << student.studentNumber << endl
       << "The quiz grades are: "
       << student.quiz1  << " " << student.quiz2
       << endl
       << "The midterm and exam grades are: "
       << student.midterm << " " << student.final
       << endl
       << "The numeric average is: " << student.average
       << endl
       << "and the letter grade assigned is "
       << student.grade
       << endl;
}
```

2)

```
//suitors.cpp
//
//This program determines where to stand in line if you would
//  like to win the hand of the princess.  The princess eliminates
//  every third suitor and loops back to the beginning of the line upon
//  reaching the end.
//
//This program uses a vector to store the list of suitors and removes
//  each one in turn.
#include <iostream>
#include <cstdlib>
#include <vector>
using namespace std;
// ======================
//     main function
```

```
// ======================
int main()
{
  // Variable declarations
  int i;
  int current;
  int numSuitors;
  cout << "Enter the number of suitors" << endl;
  cin >> numSuitors;
  vector<int> suitors(numSuitors);
for (int i=0; i<numSuitors; i++)
  {
  suitors[i] = i+1; // Number each suitor's position
  }
// --------------------------------
// ----- ENTER YOUR CODE HERE -----
// --------------------------------
  if (numSuitors <=0)
  {
  cout << "Not enough suitors." << endl;
  }
  else if (numSuitors == 1)
  {
  cout << "You would stand first in line." << endl;
  }
  else
  {
  current=0;  // Current suitor the princess will examine
  // Eliminate a suitor as long as there is at least one
  while (suitors.size() > 1)
  {
```

```
 // Count three people ahead, or go two people down
 // since we include the current person in the count
 for (i=0; i<2; i++)
 {
 current++;
 // If we reached the end, go back to the front
 if (current == suitors.size())
 {
 current=0;
 }
 }
 // Eliminate contestant current
 suitors.erase(suitors.begin() + current);
 // If we were at the last suitor, go to the first one
 if (current == suitors.size())
 {
  current=0;
 }
   }
   cout << "To win the princess, you should stand in position " <<
 suitors[0] << endl;
 }

// --------------------------------
// --------- END USER CODE --------
// --------------------------------
 return 0;
}
```

3)

```
//Ch6prg3.cpp
#include <iostream>
```

```
using namespace std;
//  Point
//  The members should implement
// a)a member function, set, to set the private data after creation
// b)a member function to move the point a vertical distance and a
//   horizontal distance specified by the first and second arguments.
// c)a member function that rotates the point 90 degrees clockwise
//   about the origin.
// d)two const inspector functions to retrieve the current coordinates
//   of the point.
//  Document the member functions.
//  Test with several points exercise member functions.
class Point
{
public:
  //set: set x to first, y to second
  void set(int first, int second);

  //move point horizontally by distance first
  //move vertically by distance second
  void move(int first, int second);
  //rotate point 90 degrees clockwise
  void rotate();

  // returns the first coordinate of the point
  double first();
  // returns the second coordinate of the point
  double second();
private:
  double x;
  double y;
```

```
};
double Point::first()
{
  return x;
}
double Point::second()
{
  return y;
}
void Point::set(int first, int second)
{
  x = first;
  y = second;
}
void Point::move(int first, int second)
{
  x = x + first;
  y = y + second;
}
void Point::rotate()
{
  double tmp = x;
  x = -y;
  y = tmp;
}
int main()
{
  Point A, B, C;
  A.set(1,2);
  cout << A.first() << ", " << A.second() << endl;
  A.rotate();
```

```
  cout << A.first() << ", " << A.second() << endl;
A.rotate();
  cout << A.first() << ", " << A.second() << endl;
  A.rotate();
  cout << A.first() << ", " << A.second() << endl;
  A.rotate();
  cout << A.first() << ", " << A.second() << endl;
  A.rotate();
  cout << A.first() << ", " << A.second() << endl;
  B.set(2,3);
  cout << B.first() << ", " << B.second() << endl;
  B.move(1,1);
  cout << B.first() << ", " << B.second() << endl;
  C.set(5, -4);
  cout << C.first() << ", " << C.second() << endl;
  cout << "Move C by -5 horizontally and 4 vertically. " << endl;
  C.move(-5, 4);
  cout << C.first() << ", " << C.second() << endl;
  return 0;
}
```

4)

```
// pizza.h
//
// Interface file for the Pizza class.
const int SMALL = 0;
const int MEDIUM = 1;
const int LARGE = 2;
const int DEEPDISH = 0;
const int HANDTOSSED = 1;
const int PAN = 2;
class Pizza
```

```
{
  public:
    Pizza();
    ~Pizza() {};
    int getPepperoniToppings();
    void setPepperoniToppings(int numPepperoni);
    int getCheeseToppings();
void setCheeseToppings(int numCheese);
    int getSize();
    void setSize(int newSize);
    int getType();
    void setType(int newType);
    void outputDescription();
    double computePrice();
  private:
    int size, type, pepperoniToppings, cheeseToppings;
};
// pizza.cpp
//
// This program implements the Pizza class and creates several
// pizza objects to test it out.
#include <iostream>
#include "pizza.h"
using namespace std;
//=======================
// Pizza
// The constructor sets the default pizza
// to a small, deep dish, with only cheese.
//=======================
Pizza::Pizza()
{
```

```
  size = SMALL;
  type = DEEPDISH;
  pepperoniToppings = 0;
  cheeseToppings = 1;
}
//==================================
// Accessors and Mutators Follow
//==================================
 // --------------------------------
 // ----- ENTER YOUR CODE HERE -----
 // --------------------------------
int Pizza::getPepperoniToppings()
{
   return pepperoniToppings;
}
void Pizza::setPepperoniToppings(int numPepperoni)
{
   pepperoniToppings = numPepperoni;
}
int Pizza::getCheeseToppings()
{
   return cheeseToppings;
}
void Pizza::setCheeseToppings(int numCheese)
{
   cheeseToppings = numCheese;
}
int Pizza::getSize()
{
   return size;
}
```

```
void Pizza::setSize(int newSize)
{
  size = newSize;
}
int Pizza::getType()
{
  return size;
}
void Pizza::setType(int newType)
{
  type = newType;
}
//==================================
// outputDescription
// Prints a textual description of the contents of the pizza.
//==================================
void Pizza::outputDescription()
{
  cout << "This pizza is: ";
  switch (size)
  {
    case SMALL:  cout << "Small, ";
 break;
    case MEDIUM:  cout << "Medium, ";
 break;
    case LARGE:  cout << "Large, ";
 break;
    default: cout << "Unknown size, ";
  }
  switch (type)
  {
```

```
   case DEEPDISH: cout << "Deep dish, ";
break;
   case HANDTOSSED: cout << "Hand tossed, ";
break;
   case PAN: cout << "Pan, ";
break;
   default: cout << "Uknown type, ";
  }
  cout << "with " << pepperoniToppings << " pepperoni toppings " <<
      "and " << cheeseToppings << " cheese toppings." << endl;
}
//==================================
// computePrice
// Returns:
//   Price of a pizza using the formula:
//     Small = $10 + $2 per topping
//     Medium = $14 + $2 per topping
//     Large = $17 + $2 per topping
//==================================
double Pizza::computePrice()
{
  double price = 0;
  switch (size)
  {
    case SMALL: price = 10; break;
    case MEDIUM: price = 14; break;
    case LARGE: price = 17; break;
    default: cout << "Error, invalid size." << endl;
     return -1;
  }
  price += (pepperoniToppings + cheeseToppings) * 2;
```

```
  return price;
}
 // --------------------------------
 // --------- END USER CODE --------
 // --------------------------------
// ======================
//     main function
// ======================
int main()
{
 // Variable declarations
 Pizza cheesy;
 Pizza pepperoni;
 cheesy.setCheeseToppings(3);
 cheesy.setType(HANDTOSSED);
 cheesy.outputDescription();
 cout << "Price of cheesy: " << cheesy.computePrice() << endl;
 pepperoni.setSize(LARGE);
 pepperoni.setPepperoniToppings(2);
 pepperoni.setType(PAN);
 pepperoni.outputDescription();
 cout << "Price of pepperoni : " << pepperoni.computePrice() << endl;
 cout << endl;
}
***
```