EECE 230 Introduction to Programming
Quiz I

March 5, 2013

- The duration of this exam is 3 hours.
- It consists of 5 problems.
- You will have 15 minutes at the end of the exam to upload your answers to the moodle website. It is your responsibility to make sure your files are correctly uploaded.
- You can use the C++ lectures and the C++ tutorial in pdf format on your lab machines.
- You are **NOT** allowed to use the **web** (**imail** included). You are not allowed to use **USB's** or files previously stored on your machine.
- If you violate the above rules or if you communicate with a person other than the exam proctors during the exam, you will immediately get zero and you will be referred to the appropriate disciplinary committee.
- Active cell phones and any other unauthorized electronic devices are absolutely not allowed in the exam rooms. They should be turned off and put away.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Submit your solutions each part in a separate file as indicated in the booklet. Include your name and ID number in each file.
- Good luck!

Problem 1 (25 points). Mine field.

The coordinates x and y in the following equation give the points of an ellipse with radii a and b centered at (x_0, y_0).

$$(\frac{x - x_0}{a})^2 + (\frac{y - y_0}{b})^2 = 1$$

You discovered an enemy location at position $(1.2, 2.2)$. The location is protected by an elliptical mine field of radii a and $2.5 \times a$. Your friends are all placed parallel to the x axis. Help them avoid the mines.

- Take the value of a from an informant user.
- Assume $b = 2.5 \times a$.
- Take the x coordinates from your friends one at a time.
- Send back (print) the y coordinates of the mines that correspond to the x coordinates passed by your friends.
- If no mines exist along x, print "safe to cross".
- Your program should run until 5 of your friends cross safely.

Sample input/output:

```
Dear informant, please enter 'a': 1

Dear friend 1, please enter 'x': 10
safe to cross.

Dear friend 2, please enter 'x': 1.8
avoid points with y between 4.2 and 0.2
```

Correct and detailed comments are worth **7 points**. Compiling and working solutions get **10 points**. The remaining **8 points** are for style, correctness, indentation, thinking and problem solving skills.

Submit your solution in a file called Prob1.cpp including your name and ID number.

Problem 2 (15 points). Intersecting intervals.

Write a program that takes the real numbers x_1, x_2, y_1 and y_2 from the user. An interval $I = [x_1, x_2]$ contains all real numbers between x_1 and x_2. The interval $J = [y_1, y_2]$ contains all real numbers between y_1 and y_2. Your program should print whether I and J intersect or not.

Correct and detailed comments are worth **4 points**. Compiling and working solutions get **7 points**. The remaining **4 points** are for style, correctness, indentation, thinking and problem solving skills.

Submit your solution in a file called Prob2.cpp including your name and ID number.

Problem 3 (25 points). In-order insertion.

The following code initializes an array a of capacity 256 with 16 in-order elements.

- Write code to take an additional element e from the user, and insert e in its correct position in a so that a is still ordered. Update the necessary variables.

- Keep taking additional elements from the user and inserting them in place until the user enters a number that already exists in a, a number larger than the maximum in a or a number smaller than the minimum in a.
- Your code should be as efficient as possible.

```
#include <iostream>//needed for cin, cout, endl
using namespace std;
const int CAPACITY = 256;

int main() {
  int a[CAPACITY] = {
     2, 4, 7, 8,10,18,24,32,
    33,34,41,51,66,67,80,90};
  int n = 16;
  int e = 0;
  //write code here to get e from the user
  // and insert it in place ...
  return 0;
}
```

A sample run is provided below.

```
Please enter an element: 12
Array now is:
{2,4,7,8,10,12,18,24,32,33,34,41,51,66,67,80,90}

Please enter an element: 18
Element 18 already exists.
Exiting.
```

Correct and detailed comments are worth **5 points**. Compiling and working solutions get **10 points**. More efficient solutions are worth up to **5 points**. The remaining **5 points** are for style, correctness, indentation, thinking and problem solving skills.

Submit your solutions in a file called Prob3.cpp including your name and ID number.

Problem 4 (35 points). Electricity Bills.

Electricity bills report the number of kilowatt hours (KWH) used by a customer. We determine the KWH amount consumed in the current month by computing the difference between the current meter reading and the meter reading of the previous month. For example, if the meter in January read 25,000 KWH, and it read in February 27,050, then 27,050-25,000 = 2,050 KWH were consumed. The meter has only 5 digits. It rolls back to 00,000 directly after 99,999.

- **5 pts.** Write a program to take n readings from the user and store them in an array.
- **15 pts.** Compute the index of the month with the peak power usage. Be as efficient as you can.
- **15 pts.** Compute and print the bill for every month. Note that the first 1,000 KWH are billed at 5 cents per KWH. Then the next 750 KWH are billed at 10 cents per KWH. Then the next KWHs are billed at 25 cents per KWH. Compute the cost in cents, and then report it to the user in dollars.

An example sample run is provided below.

```
Please enter the number of meter readings: 6
Enter the meter readings one at a time:
10550
11350
```

```
14500
15250
16000
18020
The index of the peak KWH consumption is 2: 14500 - 11350 = 3150
price for month 1: 800*5 + 0*10 + 0*25=4000cents = 40 dollars.
price for month 2: 1000*5 + 750*10 + 1400*25=47500cents = 475 dollars.
price for month 3: 750*5 + 0*10 + 0*25=3750cents = 37 dollars.
price for month 4: 750*5 + 0*10 + 0*25=3750cents = 37 dollars.
price for month 5: 1000*5 + 750*10 + 270*25=19250cents = 192 dollars.
```

Detailed and correct comments are worth **6 points**. Error and boundary conditions are worth **6 points**.

Submit your solutions in a file called Prob4.cpp including your name and ID number.

Problem 5 (40 points). Estimate integrals using pseudo-random numbers.

Random processes observe possibilities with uniform probabilities. For example, tossing a coin, one expects head and tail with equal probability. Pseudo-random number generators take a seed from the user and generate a sequence of numbers that appear to be random.

The function `srand` takes a seed number and should be called once before calling `rand`. A sequence of calls to the function `rand` generates pseudo-random integer numbers between 0 and `RAND_MAX`. The functions `srand` and `rand` and the constant `RAND_MAX` are all defined in the `stdlib` library.

Write a program that performs the following **10 pts.**

- Obtain a seed from the user and pass it to `srand`. Calling `srand` with different seeds in separate runs of the program, guarantees generating different sequences.
- Call `rand` and use arithmetic operations like division, multiplication, addition, and remainder to compute a pseudo-random real number x_1 between 0 and 2.
- Call `rand` and use arithmetic operations like division, multiplication, addition, and remainder to compute a pseudo-random real number y_1 between 0 and 3.

Consider the integral $\int_0^2 \sqrt{9-x^2}\,\mathrm{d}x$. Perform the following tasks **30 pts.**

- Check whether (x_1, y_1) is under the curve $\sqrt{9-x^2}$.
- Generate 10,000 points $(x_i, y_i), 1 \leq i \leq 10,000$ in a similar manner (inside the box $(0, 2, 0, 3)$) and count how many points end up being under the curve $\sqrt{9-x^2}$.
- Print the ratio of the points under the curve to the total number of points. That should be a good estimate of the area under the curve $\sqrt{9-x^2}$ and in fact a solution to the integral.

Detailed and correct comments are worth **7 points**.

Submit your solution in a file called Prob5.cpp including your name and ID number.