# EECE 230 Introduction to Programming, Sections 3,4, and 12
# Final Exam

January 8, 2013

- The duration of this exam is 3 hours.
- It consists of 5 problems.
- The exam is open moodle. You can use all the material on Moodle: lecture notes, programming assignments, and solutions, etc. You are **NOT** allowed to use the **web** (**imail** included). You are not allowed to use **USB's** or files previously stored in your **account**.
- 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 a 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. Submit the files online in a single zip file called *abcMN.zip*, where *abcMN* is your AUB user i.d., e.g., *abc*01.*zip*.
- Good luck!

**Problem 1 (10 points). Number of zeros**

Write a program which given a list of integers, finds the number of zeros in the input list. Assume that the list end is indicated by the sentinel -999.

*Examples:*

- The number of zeros in the sequence $1, 10, 3, -5, -5, 10, -5, -1, -999$ is 0
- The number of zeros in the sequence $1, 10, 0, 3, -5, -5, 10, -5, -1, -999$ is 1
- The number of zeros in the sequence $1, 10, 0, 3, -5, -5, 0, 10, -5, -1, -999$ is 2
- The number of zeros in the sequence $1, 0, 0, 3, -5, -5, 0, 10, -5, -1, -999$ is 3.

Submit your code in a file called Prob1.cpp. Include your name and ID number in the file.

**Problem 2 (20 points). Check if all the integers are repeated the same number of times**

Write a program which takes as input a list of integers whose end is indicated by the sentinel $-999$. Your program is supposed to check whether or not all the integers in the list appear the same number of times. If the answer is YES, your program is supposed to compute the number $R$ of times each integer is repeated.

*Examples:*

- If the input list is

```
1 10 2 3  -999
```

  then the answer is YES since each integer is repeated once, hence $R = 1$.

- If the input list is

```
10  1 10 2 2  3  1 3  -999
```

  then the answer is YES and $R = 2$.

- If the input list is

```
1 2 10  1 10 2 10 2  3  1 3 3  -999
```

  then the answer is YES and $R = 3$.

- If the input list is

```
10  1 10 2  3  1 3  -999
```

  then the answer is NO since, for instance, 2 appears once and 1 twice.

- If the input list is

```
1 10 2 10 3  -999
```

  then the answer is NO since, for instance, 10 appears twice and 1 once.

Faster programs are worth more points. Any correct solution of the problem is worth 15/20 points.
Submit your code in a file called Prob2.cpp. Include your name and ID number in the file.

**Problem 3 (25 points). Classes: fraction class**

In this problem we are interested in *fractions*, i.e., numbers of the form $a/b$ where $a$ and $b$ are integers such that $b \neq 0$. We don't want to loose accuracy by storing such numbers in the float or double format.

Design a class $fraction$ that defines a fraction as an Abstract Data Type.

Include the member functions:

- Default constructor which sets the fraction to 0/1.
- A constructor which takes $a$ and $b \neq 0$ as input parameters. The program must exit if $b = 0$.
- Print member function to print the fraction in the format $(a/b)$.
- The member function *add* to perform the addition of two fractions. For instance, consider

```
fraction x(1,2);
fraction y(2,3);
fraction z;
z = x.add(y);
```

  Then $z = 1/2 + 2/3 = 7/6$.

- The member function *mult* to perform the multiplication of two fractions. For instance, in the above example the statement $z = x.mult(y)$ results in setting $z = (1/2) * (2/3) = 2/6$.
- The member function *neg* to perform the negation of a fractions. For instance, in the above example the statement $z = x.neg()$ results in setting $z = -1/2$.
- The member function *isEqual* to check whether or not two fractions are equal. For instance, in the above example the statement $x.isEqual(y)$ returns $FALSE$.

  Be careful: $1/2 = 2/4$.

- The member function $simplify$ to simplify a fraction. For instance, consider

```
fraction u(120,180);
fraction v;
v = u.simplify();
```

  Then $v = 2/3$.

Write a program to test your class. Your test program is worth a part of the grade.

Submit this problem in a file called Prob3.cpp. Include your name and ID number in the file.

**Problem 4 (20 points). Similar strings**

In this problem we are interested in $C$-strings which are either the same or similar in the sense that one can be obtained from the other by deleting or inserting a character or swapping two adjacent characters.

a) **(10 points) Delete or insert.** Write a function

```
bool similarA(char *s, char * t)
```

which given two C-string $s$ and $t$, checks whether or not $s$ and $t$ are either exactly the same or $t$ can be obtained from $s$ by deleting a character or inserting a character.

*Examples:*

- If $s =$ "*problem*" and $t =$ "*problem*", the answer is YES (they are the same).
- If $s =$ "*problem*" and $t =$ "*proble*", the answer is YES (delete $m$)
- If $s =$ "*problem*" and $t =$ "*prolem*", the answer is YES (delete $b$).
- If $s =$ "*problem*" and $t =$ "*probslem*", the answer is YES (insert $s$).
- If $s =$ "*problem*" and $t =$ "*probleme*", the answer is YES (insert $e$).
- If $s =$ "*problem*" and $t =$ "*proleme*", the answer is NO (we can either delete or insert but not both).
- If $s =$ "*problem*" and $t =$ "*poblm*", the answer is NO (we can delete at most one character)
- If $s =$ "*problem*" and $t =$ "*abcm*", the answer is NO

b) **(10 points) Swap.** Write a function

```
bool similarB(char *s, char * t)
```

which given two C-string $s$ and $t$, checks whether or not $s$ and $t$ are either exactly the same or $t$ can be obtained from $s$ by swapping two adjacent characters.

*Examples:*

- If $s =$ "*problem*" and $t =$ "*problem*", the answer is YES (they are the same).
- If $s =$ "*problem*" and $t =$ "*porblem*", the answer is YES (swap $r$ and $o$).
- If $s =$ "*problem*" and $t =$ "*rpoblem*", the answer is YES (swap $p$ and $r$).
- If $s =$ "*problem*" and $t =$ "*problme*", the answer is YES (swap $e$ and $m$).
- If $s =$ "*problem*" and $t =$ "*rpoblme*", the answer is NO (only one swap is allowed)
- If $s =$ "*problem*" and $t =$ "*proble*", the answer is NO
- If $s =$ "*problem*" and $t =$ "*abcm*", the answer is NO

Write a program to test your functions. Use the above examples.
Submit this problem in a file called Prob4.cpp. Include your name and ID number in the file.

**Problem 5 (25 points). Basketball score**

In a basketball game, a goal is worth 1 point, 2 points, or 3 points. Given the score $n$ of a team at the end of a game, we are interested in the possible ways the score $n$ can be achieved.

a) **(15 points) Count.** Write a function

```
int count(int n)
```

which given $n$, returns *the number of* possible ways the score $n$ can be achieved.

*Examples:*

i) $count(3) = 4$ since the possible scores are:

```
1 1 1
1 2
2 1
3
```

ii) $count(4) = 7$ since the possible scores are:

```
1 1 1 1
1 1 2
1 2 1
1 3
2 1 1
2 2
3 1
```

iii) $count(25) = 2555757$.

To count the possible scores, you don't have to enumerate them. You can do it more efficiently.

b) **(10 points) Enumerate.** In this part we are interested in enumerating all possible scores. Write a function

```
void enumerate(int n)
```

which given $n$, prints all possible ways the score $n$ can be achieved.

For instance *enumerate*(3) should display the possible scores shown in (i) above, and *enumerate*(4) should display the possible scores shown in (ii) above.

Note that *enumerate* takes significantly more time than *count*.

Write a program to test your functions.
Submit this problem in a file called Prob5.cpp. Include your name and ID number in the file.