CSC 320 Summer 2012: Assignment #1
Due at beginning of class, Fri. May 25

Read Chapter 1 of the text (reviews math prerequisites).

Instructions for all assignments:

1. Put your name on your assignment.
2. Assignments are submitted on paper at the beginning of class on the due date. Questions should be **in order**.
3. Show your work unless otherwise stated.
4. Draw BIG boxes for your marks on the top of the first page of your submission. Place a 0 in the corresponding box for any questions you omit. For this assignment:

Question:	1	2	3	4	5	6	7	8	9
Marks	0	0	0	0	0	0	0	0	0

1. [Notation for languages] [10] List the elements of

   (a) $A \times \phi$ (ϕ denotes the empty set),

   (b) $A \times \{e\}$ (e denotes the empty string),

   (c) $A \times B$,

   (d) $A \times A$, and

   (e) $(A \times A) \times B$,

   where A= $\{p, q\}$ and $B = \{0, 1, 2\}$.

2. [Review of algorithm time complexities][15] For this question, you must use the following definition of Big Oh:
   Let $f(n)$ and $g(n)$ be functions which map the non-negative integers into the non-negative real numbers. A function $f(n)$ is in the set $O(g(n))$ if there exists constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq c\, g(n)$.

(a) Prove that $f(n) = 2n^5 + 3n^4 + n^3 + 7n^2 + 4$ is in $O(n^5)$.

(b) Suppose that an algorithm A takes at most $k^2 + k$ steps on a problem of size k. Give a tight upper bound on the number of steps that algorithm A takes on a problem of size $2n^3 + 5$.

(c) Prove that algorithm A takes $O(n^6)$ steps on a problem of size $2n^3 + 5$.

3. [Review of induction] [15] Prove by induction that for $n \geq 1$, the number of strings of length n using the symbols a, b, and c with no two consecutive places holding the same symbol, is $3\ 2^{n-1}$. For example, there are 12 such strings of length three: aba, abc, aca, acb, bab, bac, bca, bcb, cab, cac, cba, and cbc.

4. [15] Let $S = \{w : w$ is a string over $\Sigma = \{a, b, c\}$ with no two consecutive places holding the same symbol $\}$. Is this set countable or uncountable? Prove your claim.

5. [15] Let $S = \{w : w$ is a (possibly infinite) sequence over $\Sigma = \{a, b, c\}$ with no two consecutive places holding the same symbol $\}$. Is this set countable or uncountable? Prove your claim.

[Review of graph theory, solving problems using a black box approach]

Basic definitions: A path that contains every vertex of a graph is a *Hamiltonian path*. A cycle that contains every vertex of a graph is a *Hamiltonian cycle.*
HAMILTON PATH PROBLEM: Does graph G have a Hamiltonian path?
HAMILTON CYCLE PROBLEM: Does graph G have a Hamiltonian cycle?

I've written some code for the Hamilton Path Problem and the Hamilton Cycle Problem both in C and in Java. You may select either language to do this question. Or you can use a C++ compiler with the C code (I wrote it so that it should compile as C or C++).

If you choose C, the files you require are:

1. The test driver for the program: main.c
2. The code for the functions **hamilton_path** and **hamilton_cycle** and templates for your new functions: ham.c
3. A sample input file: in.txt

Before you get started, copy these files, compile with :
gcc main.c ham.c
and run on the sample input file:
a.out < in.txt > out.txt
[on a unix machine, ask me if you do not know how to do this on your computer].

If you choose Java, the files you require are:

1. The test driver for the program (main) and the Graph class: Graph.java
2. Code to aid in the input (you do not need to know any details of this code): Keyboard.java
3. The code for the methods **hamilton_path** and **hamilton_cycle** and templates for your new methods: Hamilton.java

4. A sample input file: in.txt

Before you get started, copy these files, compile with:
javac Graph.java Keyboard.java Hamilton.java
and run on the sample input file:
java Graph < in.txt > out.txt
[on a unix machine, ask me if you do not know how to do this on your computer].
The output (out.txt) is identical for both the C and java programs.

The point of these questions is to prove that if either problem can be solved in polynomial time, so can the other. You must test your programs on the graphs in file *in.txt*. Hand in the code for your two new routines, and the output you get from testing.

6. [10] Write code for a new Hamilton Path routine **new_hamilton_path** which returns 1 if the graph it is called with has a Hamilton path and 0 otherwise. You may use my Hamilton Cycle routine **hamilton_cycle** as a black box. Explain in your comments why your algorithm always works. For full marks, your program must run in polynomial time given a **hamilton_cycle** routine that runs in polynomial time.

7. [5] Suppose that my **hamilton_cycle** routine takes time which is in $O(n^3 + m^2)$ where n is the number of vertices and m is the number of edges of the input graph. Express the time complexity of your code as a function of n and m .

8. [10] Write code for a new Hamilton Cycle routine **new_hamilton_cycle** which returns 1 if the graph it is called with has a Hamilton cycle and 0 otherwise. You may use my Hamilton Path routine **hamilton_path** as a black box. Explain in your comments why your algorithm always works. For full marks, your program must run in polynomial time given a **hamilton_path** routine that runs in polynomial time.

9. [5] Suppose that my routine **hamilton_path** takes time which is in $O(n^4 + m^3)$ where n is the number of vertices and m is the number of edges of the input graph. Express the time complexity of your code as a function of n and m.