COS 126 **General Computer Science** **Fall 2006**

Exam 2

This test has 11 questions worth a total of 50 points. You have 120 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet, handwritten by you on both sides. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Partial credit will be given for partially correct answers. **Write out and sign the Honor Code pledge before turning in the test.**
"I pledge my honor that I have not violated the Honor Code during this examination."

Signature

Problem	Score
0	
1	
2	
3	
4	
5	
Sub 1	

Problem	Score
6	
7	
8	
9	
10	
Sub 2	

Total	

Name:

NetID:

Preceptor:	Nadia	Ari
	George	Adam
	Maia	Elliott

0. **Miscellaneous. (1 point) (really)**

   (a) Write your name and Princeton NetID in the space provided on the front of the exam, and circle the name of the preceptor who grades your homework assignments.

   (b) *Write* and sign the honor code on the front of the exam.

1. **Deconstructing Hello, World. (4 points)**

   Consider the following Java program.

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World");
   }
}
```

For each code fragment on the left, find the best matching description on the right.

___ `class`

___ `public`

___ `static`

___ `void`

___ `main`

___ `String[] args`

A. Identifies the method as shared by the class, rather than one associated with each object instantiated from the class.

B. Identifies the method as not returning any value.

C. Identifies the method as available for use by any other program.

D. Identifies the method as being callable at most once within a program.

E. Identifies the method that is automatically invoked when you run `java HelloWorld`.

F. An array of strings containing the command line arguments.

G. A group of related methods and variables.

H. Has no meaning.

2. **Java programming (5 points)** Consider the following:

   - "Step on no pets!"
   - "A man, a plan, a canal: Panama!"
   - "Doc, note: I dissent. A fast never prevents a fatness. I diet on cod."
   - "Straw? No, too stupid a fad. I put soot on warts."

   All of the above are *palindromes*: they read the same forwards and backwards, if we ignore punctuation, capitalization, and spaces. In this question you will write a Java program that will test whether an English sentence is a palindrome or not.

   You will probably want to use the `String` method `toLowerCase()`, which returns a `String` with all the capital letters converted to lower-case. Write a function `isPunc(char c)` that will return `true` if `c` is a space or a punctuation mark (comma, period, colon, semicolon, exclamation point, question mark). Your program should behave as follows:

```
% java Palindrome "Step on no Pets!"
true
%java Palindrome "Flo, gin is a sin; I play golf."
false
```

   In other words, the `String` will be quoted, so `args[0]` will contain the whole thing. Flll in the spaces below with your code (we've begun `isPunc` for you):

```
public class Palindrome {

   public static boolean isPunc(char c){
        if ((c == ',') ||


   }

   public static void main(String[] args) {










    }
}
```

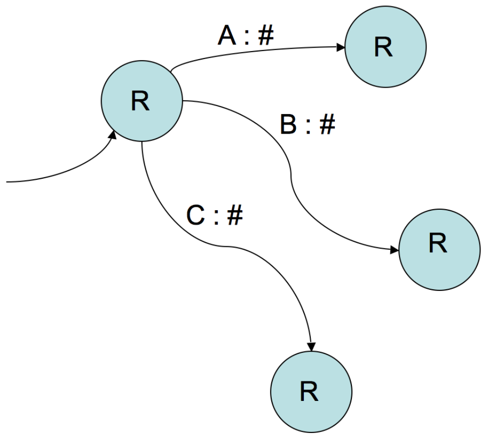
3. **Turing Machines (4 points)**

   Now complete the design below for a Turing Machine for the palindrome problem, but to keep things reasonable, the input you'll check will be a sequence of consecutive A's and B's and C's written between # characters, with no punctuation or spaces.

   You may change the input sequence in any way you like, including erasing the whole thing. Just before it halts, your machine should print a "Y" if the input sequence was a palindrome (for example, ABACCCABA), and an "N" if it was not (for example, BABBCC).

   Assume that the initial position of the Turing Machine head is at the leftmost A, B, or C of the input string. Remember that the very first thing a Turing Machine does when started in its initial state is inspect the symbol under the head—it does not move first!

   While any correct solution will receive full credit, at most five more states (one of them the Halt state) are needed, in addition to those shown below.



4. **Circuits (4 points)**

   Now design a logic circuit that will detect binary palindromes exactly 9 bits wide. (For example, 000101000 and 101000101 are 9-bit palindromes, but 111100111 and 010101011 are not.) The inputs to your circuit are the 9 bits. The 1-bit output should be 1 if the input is a palindrome, and 0 if it is not. Use any number of the following parts in any combination:

   - AND and OR gates with any number of inputs,
   - NOT gates,
   - XOR (exclusive OR) gates with just two inputs, and of course
   - wires.

   You may represent your gates in any way you like. Boxes with labels will be just fine.

   Hint: Of course you could build a 512-line truth table and design an absolutely gigantic circuit, but there are much simpler solutions. Also, unlike what you did in the last two questions, here you'll be checking all of the bits simultaneously.

   Once again, while any correct solution will receive full credit, this can be done with just six gates from the above list (and a few wires).

5. **Linked structures (6 points)**

A doubly-linked list is a kind of linked list in which each node not only has a pointer to the next node in the list but also has a pointer to the previous node in the list. A partial implementation of a doubly-linked list is shown below. You will implement one method.

```
public class DoublyLinkedList
{
    private Node first;     // the first Node in the list
    private Node last;      // the last Node in the list

    private class Node
    {
        private Point p;
        private Node prev;  // the previous Node
        private Node next;  // the next Node
    }

    . . . methods go here . . .

}
```

You can assume the following:

- When the list is empty, `first == null` and `last == null`.
- When the list is not empty, `first.prev == null` and `last.next == null`.
- The `Point` objects are of the same type as those used in the TSP assignment.

In the space below, implement the method `public void reverse()` of `DoublyLinkedList` that reverses the order of the entire list. Your implementation should iterate through the entire list no more than once.

6. **Abstract Data Types (6 points)**

   You've seen how the Queue ADT can be implemented with a linked list. In this question you'll instead implement Queue (of Strings) with an array. Fill in the blanks in the code below (one constructor, three methods). Note that the constructor takes a parameter that is the maximum size of the queue. You may assume that the number of Strings in the queue will never be as large as this. But the amount of enqueueing and dequeueing by the client is not limited. You may want to draw a picture of the array, thinking carefully about what should happen when `front` or `back` approaches `N`. You may assume that the client will never dequeue from an empty queue.

```
public class ArrayQueue  {

   private String [] a;
   private int N;          // the size of the ArrayQueue array
   private int back;      // where to put the next String to be enqueued
   private int front;     // where to get the next String to be dequeued

   public ArrayQueue(int max) {



   }

   public boolean isEmpty() {



   }

   public void enqueue(String s) {




   }

   public String dequeue() {




   }
}
```

7. **True/False. (6 points)**

   Write T for true or F for false next to each of the following statements, according to their veracity.

   ___ A recursive function can always be converted into an equivalent non-recursive one.

   ___ A `while` loop can always be converted into an equivalent `for` loop, and vice versa.

   ___ The regular expression `a*b*` represents the set of all strings consisting of `a`'s and `b`'s.

   ___ The undecidability of the halting problem is a statement about Turing machines; it is *not* applicable to real computers.

   ___ P is the class of search problems for which a polynomial-time Java program could, in principle, be written.

   ___ NP is the class of search problems for which *no* polynomial time Java program could ever be written.

   ___ NP is the class of search problems for which, in principle, a Java program could be written to check a given proposed solution in polynomial time.

   ___ If you discover a polynomial-time algorithm for any problem in NP, then all problems in NP are solvable in polynomial-time.

   ___ The NP complete problems are the hardest members of NP.

   ___ Since FACTOR is a problem in NP, any instance of FACTOR can be restated as an instance of 3-SAT (3-satisfiability).

8. **Object-Oriented Programming (6 points)**

Consider the following API, which describes intervals on the real line. An interval is defined to be the set of all `double`s greater than or equal to `left` and less than or equal to `right`.

```
public class Interval
---------------------------------------------------------------------------
        Interval (double left, double right)
boolean contains(double x)                 is x in this interval?
boolean intersects(Interval b)             do this interval and b intersect?
 String toString()                         string representation
```

Do NOT implement this data type! Instead, making use of the Interval API, and using the space below, write a client that takes an `int` value `N` as a command–line argument, reads `N` intervals (each represented by a pair of `double` values) from standard input, and prints all `Interval`s that intersect. Any clear output format is fine.

9. **TigerSoft (6 points)** It is the year 2019. After graduating from Princeton, you went to Harvard Business School and got your MBA degree. You then went to work for the TigerSoft software company, and due to your talent and ambition (not to mention your superior education), you are now Chief Software Architect at the company. You provide technical leadership and vision to a large team of programmers and also advise senior management on technical challenges and opportunities.

   All is not well at TigerSoft. Your team of programmers turns out great quantities of software, but the programs are typically full of subtle bugs, resulting in unhappy customers, loss of revenue, and a decline in the value of your stock options. The Chief Executive Officer of TigerSoft, also a Princeton alum (but who somehow never took COS 126) is very upset and seeks your advice. The CEO would like you to institute a Quality Assurance effort along the following lines.

   For each piece of software the responsible programmer will supply a list of some (but, of course, not all) possible inputs and a matching list of the correct outputs, which the programmer will figure out by hand. The CEO would like you to write a special program, to be called the Verifier, which will automatically check that each new piece of software behaves according to its list of inputs and matching outputs. The Verifier will accept as its own input an arbitrary function `f()` to be tested (a function that computes cosines, for example), one possible set of input data for `f()` (an angle, in this example), and the correct output for that input (the correct cosine, in this example). The Verifier will then check whether or not `f()` will eventually produce the correct output given that input, and then report the result by printing "Yes" or "No". The Verifier can be used to check all the inputs on the programmer-supplied list simply by running it over again for each new input/output combination.

   How would you advise the TigerSoft CEO on this idea? Can such a Verifier be developed? If so, what would be its basic structure? If not, why not?

   The CEO, while clever, has a short attention span, so you'll need to confine your answer to one page (the following one).

9. **(continued)**

MEMO

TO: Most Worthy TigerSoft CEO

FROM: Humble Chief Software Architect

RE: Your idea for software Verifier

10. **Surprise (2 points)**

    What did you learn in this course that greatly surprised you? (Full points for any serious answer.) Please use no more than half the space below for your answer.