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. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. 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

Name:

NetID:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Sub 2</td>
<td></td>
</tr>
<tr>
<td>Sub 1</td>
<td></td>
<td>Sub 2</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P01  TTh 1:30  Andrea
P01A TTh 1:30  Sid
P01B TTh 1:30  Woo Chang
P02  TTh 2:30  Forrest
P02A TTh 2:30  Ananya
P03  TTh 3:30  Chang
P04  TTh 7:30  Tim
P05  WF 10   10   Yaping
P06  WF 11   Maia
P07  WF 1:30  Ganesh
P07A WF 1:30  Sonya
P07B WF 1:30  Yi
0. Miscellaneous. (1 point)
   (a) Write your name and Princeton NetID in the space provided on the front of the exam, and circle your precept number.
   (b) Write and sign the honor code on the front of the exam.

1. Data types. (6 points)
   (a) Define what is a data type.

   (b) For each entry on the left, find the best matching description on the right. Use each choice once.

   _ _ _ A data type whose representation is hidden from the client.
   A. Reference type.
   B. Primitive type.
   C. Abstract (encapsulated) data type.
   D. Immutable data type.

   _ _ _ After constructing an object of this type, you cannot change its value.

   _ _ _ int

   _ _ _ When an object of this type is passed to a function, the function can, in general, change its value.

   (c) List 3 compelling reasons why experienced programmers create and use data types.

   •

   •

   •
2. **Algorithms and data structures. (5 points)**

For each term on the left, find the best matching algorithm/data structure on the right.

- ___ Implement recursion.  
  - A. Stack  
- ___ Parse an NCBI genome data file.  
  - B. Queue  
- ___ Evaluate an arithmetic expression.  
  - C. Symbol table  
- ___ Implement the back button in a browser.  
  - D. Graph  
- ___ Model pairwise relationships in facebook.  
  - E. Regular expression  
- ___ Search for an IP address given a domain name.  
- ___ Model the buffer in the Karplus-Strong algorithm for simulating the pluck of a guitar string.

3. **Floating point precision. (2 points)**

Your colleague needs to compute the function $f(x) = (1 - \cos(x))/x^2$, and implements it as following:

```java
public static double g(double x) {
    return (1.0 - Math.cos(x)) / (x*x);
}
```

It returns an accurate answer for many values of $x$, but it returns a highly inaccurate answer when $x$ is close to zero, say $1.1e-8$. Explain why. (Note that $\cos x \approx 1 - x^2/2$ when $x$ is close to zero.)
4. Creating data types. (4 points)

Consider the following buggy implementation of a data type for 3-dimensional points. Fix all of the errors.

```java
public class Point3D {
    private double x, y, z;

    // create a point (x0, y0, z0)
    public void Point3D(double x0, double y0, double z0) {
        double x = x0;
        double y = y0;
        double z = z0;
    }

    // return the Euclidean distance between this point and q
    public static double distanceTo(Point3D q) {
        double dx = x - q.x;
        double dy = y - q.y;
        double dz = z - q.z;
        return Math.sqrt(dx*dx + dy*dy + dz*dz);
    }

    // return a string representation of this point
    public String toString() {
        String s = "(" + x + ", " + y + ", " + z + ")";
        System.out.println(s);
    }

}
```
5. Analysis of algorithms. (4 points)

(a) Suppose that you run the ClosestPair program on a variety of inputs and tabulate the input size versus running time.

<table>
<thead>
<tr>
<th>( N )</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000</td>
<td>1.1</td>
</tr>
<tr>
<td>10,000</td>
<td>2.2</td>
</tr>
<tr>
<td>20,000</td>
<td>9.4</td>
</tr>
<tr>
<td>40,000</td>
<td>36.1</td>
</tr>
<tr>
<td>80,000</td>
<td>140.5</td>
</tr>
<tr>
<td>160,000</td>
<td>557.5</td>
</tr>
</tbody>
</table>

Predict (within 10%) how long it will take (in seconds) to solve an instance with 800,000 points. Circle your answer.

(b) Now, suppose that you have been able to devise a more clever algorithm for the problem.

<table>
<thead>
<tr>
<th>( N )</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000</td>
<td>1.0</td>
</tr>
<tr>
<td>10,000</td>
<td>2.2</td>
</tr>
<tr>
<td>20,000</td>
<td>4.7</td>
</tr>
<tr>
<td>40,000</td>
<td>10.1</td>
</tr>
<tr>
<td>80,000</td>
<td>21.5</td>
</tr>
<tr>
<td>160,000</td>
<td>42.6</td>
</tr>
</tbody>
</table>

Predict (within 10%) how long it will take to solve an instance with 800,000 points using this algorithm. Circle your answer.
6. **Linked structures. (5 points)**

Suppose that you have a null-terminated linked lists of 3-dimensional points (Point3D from Question 4), where each element in the linked list is of type:

```java
private class Node {
    private Point3D p;
    private Node next;
}
```

Given an instance variable first that points to the first Node in the linked list, write a method `distance()` that returns the total length of the path. You may assume that the list is not empty.

For example, if the 4 points in the list are (2.0, 0.0, 3.0), (4.0, 0.0, 6.0), (1.0, 2.0, 1.0), and (3.0, 3.0, 1.0), the distance is $\sqrt{13} + \sqrt{25} + \sqrt{5}$.

```java
public double distance() {

}
```
7. Modular programming. (5 points)

Complete the implementation of the `Ball3D` data type for 3-dimensional balls. Use the `Point3D` from Question 4.

```java
public class Ball3D {

    // construct a ball centered at c, with radius r
    public Ball3D(Point3D c, double r) {

    }

    // does the ball contain the point p?
    // [ distance between center and p <= radius ]
    public boolean contains(Point3D p) {

    }

    // return the ball's volume = 4/3 pi r^3
    public double volume() {

    }

}
```
8. Theory of computation. (5 points)

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>A. The set of all search problems (i.e., solution can be checked in polynomial time).</td>
</tr>
<tr>
<td>Undecidable</td>
<td>B. A set of search problems that are believed to have no polynomial time solution.</td>
</tr>
<tr>
<td>Duality</td>
<td>C. A problem that cannot be solved by a Turing machine.</td>
</tr>
<tr>
<td>Church-Turing thesis</td>
<td>D. The set of all search problems that can be solved in polynomial time.</td>
</tr>
<tr>
<td>Turing machine</td>
<td>E. One machine can do any computational task.</td>
</tr>
<tr>
<td>P</td>
<td>F. If you can solve a problem in this class in polynomial time, then $P = NP$.</td>
</tr>
<tr>
<td>NP</td>
<td>G. Programs and data are each encoded as sequences of bits and can be used interchangeably.</td>
</tr>
<tr>
<td>NP-complete</td>
<td>H. Anything computable in this universe can be computed by a Turing machine.</td>
</tr>
<tr>
<td></td>
<td>I. A simple, universal, model of computation.</td>
</tr>
</tbody>
</table>
9. DNA analyzer. (8 points)

Write a complete program `AnalyzeDNA.java` (on the facing page) that takes a command-line argument \( k \), reads in a DNA string from standard input, and prints out the number of times each \( k \)-gram appears.

\[
\%
\text{more dna.txt}
\text{acataccatacat}
\%
\text{java AnalyzeDNA 1 < dna.txt}
\text{a 5}
\text{c 5}
\text{t 3}
\%
\text{java AnalyzeDNA 2 < dna.txt}
\text{ac 3}
\text{at 2}
\text{ca 2}
\text{cc 2}
\text{ct 1}
\text{ta 2}
\]

You may assume you have access to the symbol table library ST.

<table>
<thead>
<tr>
<th>public class ST&lt;Key, Val&gt; implements Iterable&lt;Key&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST()</td>
</tr>
<tr>
<td>boolean contains(Key key)</td>
</tr>
<tr>
<td>void put(Key key, Val val)</td>
</tr>
<tr>
<td>Val get(Key key)</td>
</tr>
<tr>
<td>Create an empty symbol table</td>
</tr>
<tr>
<td>is the key in the symbol table?</td>
</tr>
<tr>
<td>insert the key-value pair</td>
</tr>
<tr>
<td>return the value corresponding to the given key</td>
</tr>
</tbody>
</table>
public class AnalyzeDNA {
    public static void main(String[] args) {

        // read in the command-line argument k

        // read in the DNA string from standard input

        // create a symbol table with String keys and Integer values

        // populate symbol table with kgrams and their frequencies

        // print out kgrams and their frequencies

    }
}

You will receive partial credit for each commented chunk of code that you complete correctly.
10. Circuits. (5 points)

(a) The $\text{xnor}$ function is a boolean function of two inputs $x$ and $y$ that is true if and only if the two inputs are equal. Write the truth table for the $\text{xnor}$ function.

(b) Use the sum-of-products method to devise a boolean formula for the $\text{xnor}$ function.

(c) An $n$-bit comparator takes $2n$ inputs $(x_{n-1} \ldots x_1 x_0$ and $y_{n-1} \ldots y_2 y_1 y_0)$ and outputs true if and only the two $n$-bit integers are equal ($x_i = y_i$ for each $i$). If you were to write down the truth table for a 64-bit comparator, how many rows would it have?

(d) Construct a circuit for an 4-bit comparator using 4 $\text{xnor}$ gates and 1 (multiway) $\text{and}$ gate.