

CS 11 Fall 2000 — Mid-term

Name: _____

1. (10 points) Consider the Java code below and answer the questions that follow.

```
System.out.println("Enter a value for a: ");
int a = User.readInt();
System.out.println("Enter a value for b: ");
int b = User.readInt();

if (a < 0) {
    a = -a;
}

int i = 0;
while (i < a) {

    int k = 0;
    for (int j = 100; j >= b; j--) {
        k = j * 2;
        System.out.print(k);
        System.out.print('-');
        System.out.println(j);
    }

    k++;
    i = i + 1;

}
```

- (a) What is the output of this code if the user enters 2 for **a** and 98 for **b**?
- (b) What is the output of this code if the user enters -3 for **a** and 100 for **b**?

2. (15 points) Again, consider the Java code below. What is its output?

```
public static void main (String[] argv) {
    int x = 4;
    boolean[] y = new boolean[x];

    foo(x, y);

    System.out.println(x);
    System.out.println(y[1]);
}

public static void foo (int x, boolean[] b) {

    while (x >= 0) {
        if ((x % 2) == 1) {
            b[x] = true;
        }
        x = x - 1;
    }
}
```

3. (25 points) Provide short (at most a few sentences) answers to **any two of the three** following questions:

(a) What properties are needed for a **true** random number generator? Which one cannot be obtained by a computer, and why not?

(b) What role does a *stack* play in recursive method calls?

(c) What is the different between *in-place* and *out-of-place* sorting algorithms? Give an example of each.

4. (25 points) Write a method named `compare2D` that **compares** two 2-dimensional arrays, and returns *true* if the arrays are equivalent, *false* otherwise.

5. (25 points) Consider the following array of characters:

`Hello there!`

We can say that this array of characters *contains* the second array of characters, `“ello”`. We can also say that the first array of characters *does not contain* `“foo”`, as those characters don't appear in that order in the character array. Finally, we can say that the first array of characters *does not contain* `“Hll”`; although those letters **do** appear in that order in the character array, they do not appear contiguously.

Write a method named `contains` that determines whether or not one character array *contains* another. Specifically, this method should accept two pointers to character arrays from the caller. If the first character array contains the second, then this method should return `true`; otherwise, it should return `false`.