Binary numbers

We are used to expressing numbers in base 10, but binary numbers are base 2. Here are some simple examples of decimal numbers and their binary equivalents.

Decimal	Binary
0	00
1	01
2	10
3	11
4	100
5	101
6	110
7	111
8	1000

To convert a decimal number into binary, find the largest power of 2 (1, 2, 4, 8, 16, 32, etc.) in it. For example, let's take 17. This contains 16, which is 2 to the power of 4. Write a 1 down in the corresponding column of the binary number, as below.

2^4	2^3	2^2	2^1	2^0
1				

Then identify the next highest power of 2 in the number. For this example, that's 1 (because 17 - 2^4 = 1), so we can fill in zeros for all the powers of 2 that aren't there.

2^4	2^3	2^2	2^1	2^0
1	0	0	0	1

The same logic works to convert binary numbers to decimal ones. For this example, we'll take the binary number 10010101. The largest power of 2 is on the leftmost side; for this example, the number is 8 digits long and the highest power of 2 is 2^7, or 128. Now the next highest power is 2^4, or 16. The next is 2^2, or 4, and the last is 2^0, or 1. Adding these powers of 2 together gives us 149 for the decimal version of this number.

Try converting 3, 22, and 71 to binary; try converting 10101, 1100, and 1100101 to decimal.

Satisfaction problem

When is the following statement true?

C = (A or B) and (not A)

The trick with this question is to realize that A and B can each be either true or false; your job is to find a set of values for A and B that make the whole statement in C become true. How many possibilities do you need to check? You have 2 variables in the expression, A and B, and each of these can have 2 values, true or false. The rule is:

Number of possible combinations for A and B =

$$(\text{number of values for each variable})^{(\text{number of different variables})}$$

So there are 4 possibilities, which you can see in the table below. A look at the table below shows that only one set of values for A and B makes the statement C become true.

A	B	C
F	F	F
F	T	T
T	F	F
T	T	F

Interesting aside: if we rewrote this table for A and B with 1 for T and 0 for false, it would look like a binary number counting up from 0 to 3.

A	B
0	0
0	1
1	0
1	1

How many combinations will need to be checked for this statement?

X = (A or B) and (not C) and (not D or E)

There are 5 free variables to set, and each of them can be true or false. This gives us 2^5, or 32, combinations to check. Ugh.

Now imagine that instead of 2 values, true and false, we have 3 values: true, false, and unknown. How will this change the number of combinations for the statement X? We still have 5 free variables in X, but now we assume that each one can take on 3 values, so the number we're looking for is 3^5, or 243.

What values for A, B, and C make these statements true, assuming only true and false values are allowed? How many different combinations will you need to check?

(A and not C) or (C and not B) and B
(A or B) and (not B or C)

Finite state machines:

Design an FSM that accepts strings containing 111, and no others.

Can you design an FSM that accepts strings ending in 111? Explain how, or why not.

Algorithms

Know the difference between a polynomial-time algorithm (like Factorial) and a non-polynomial-time algorithm, like the Towers of Hanoi. Hint: how much longer will Factorial(7) take than (Factorial(6)? How much longer will it take to move a tower of 7 blocks than a tower of 6? Why might computer scientists care about this distinction?

We said in class that the halting problem isn't solvable. What does this imply about the compiler's ability to detect an infinite while loop in your Java code?

Why is every computer equivalent to a Turing machine? What parts of the Turing machine do you think might correspond to parts of the Dell computers in the lab?

Tape
Read head
Finite state machine

Moore's Law says that processor speed doubles about every 18 months. Does this imply that Turing machines, which stay the same, will eventually become bad models for computers? Defend your answer.

Explain why Factorial is recursive. Is the Tower of Hanoi algorithm recursive? What about the area program you wrote in Java? What's the basic idea behind a recursive program?

Java stuff

Know the common programming errors listed for recitation 2.

Name the variable types in Java that we've used or discussed so far. What's a simple example for each?

Consider some of the code from HelloWorld:

```
int birthyear;
```

```
System.out.println(“Enter your birthyear”);
birthyear = Integer.parseInt(stdin.readLine());
```

Explain what’s going on in the third line, in English.

What does stdin.readLine() do?

What potential problem can happen in a while loop, but not in an if statement? How likely is it that you will see this problem somewhere on the exam?

Given a boolean variable g, and two integer variable a = 3 and b = 1, what will g’s value be after the following Java code assignment:

```
g = ((a > 0 && b < 10) || (b > 0 && a % b != 2));
```

What is the value of q after the following code?

```
int q = 0;
while (q < 10)
    q = q + 3;
```

What about here?

```
int q = 0;
while (q < 10)
    while (q != 25)
        q = q + 1;
```

Or here?

```
int q = 0;
if (q < -1)
    while (q > -30)
        q = q – 2;
else
    while (q <= 10)
        q = (q + 1) * (q + 2);
```

Or here?

```
int q = 9;
int m = 2;
while (2 * m + q % m < 100)
    q = q % m + 25;
```

Is this code going to work, or not? Why?

```
int q = 0;
while (q = 3)
        System.out.println(q);
```

What’s this print to the screen?

```
int a = 2;
int b = 9;
System.out.println(“Answer is “ + (a + b * 2) + “ ducks”);
```