LAB 3

Adding and Multiplying

EXPERIMENT 3.1. On paper, add 658 and 986:

(a) Using the usual way, writing one under the other and carrying 1's where necessary.

(b) In a more lazy way, making a new number out of the carried 1's and adding it in separately (and repeating the lazyness as necessary).

EXPERIMENT 3.2. Using the lazy approach, calculate:

(a) $51210 + 37019$

(b) $45780 + 73175$

EXPERIMENT 3.3. Use Mathematica's `BaseForm` command to view 658 and 986 in binary (the way they are actually stored on the computer).

(a) Staying in binary, add the two numbers while carrying 1's.

(b) Staying in binary, add the two numbers using the lazy approach.

(c) Each step of the lazy approach produces two numbers. Explain concisely how each digit of the *partial sum* and the *carry* is produced.

(d) Add 110101111_2 and 1101101000_2 using lazy addition.

REMARK. Lazy addition in binary is actually the way that computers add numbers. It's extremely fast.

EXPERIMENT 3.4. Make a command `WhileDemonstrationFunction`:

```
WhileDemonstrationFunction[x_] := Module[{a = x, b = 0},
  While[a > 0,
   a = a - 1;
   b = b - 1;
   ];
   b
]
```

Plug some numbers into WhileDemonstrationFunction. What does it do? Make sure you understand how it works.

REMARK. For the rest of the worksheet, forget about negative numbers.

PROBLEM 3.5. Write a function called `SlowMultiply` that multiplies two numbers using the algorithm from class. Make sure it works, and then use the `Timing` command to compare its speed with Mathematica's built in multiplication.

EXPERIMENT 3.6. (a) Which of the following is divisible by 10? If it's divisible by 10, divide it. Why is this so easy?

$$8389, 6665, 1730, 5578$$

(b) Which of the above numbers is divisible by 2? If it's divisible by 2, divide it.

(c) Which of the following is divisible by 2? If it's divisible by 2, divide it (leaving it in base-2).

$$1010101000_2, 1101010101_2, 100001010_2, 111010000_2$$

PROBLEM 3.7. What does each of the following functions do?

(a)
```
fun[x_]:=If[x==0, "It's zero", "It's not zero"]
```

(b)
```
func[x_]:=x*func[x-1]
```

(c)
```
funct[x_]:=If[x==0, 1, x*funct[x-1]]]
```

PROBLEM 3.8. Using the fact that it's so easy for a computer to divide by 2, write a new function `FastMultiply`. How does its speed compare to `SlowMultiply`?