LAB 8

# MyPowerMod

Problem 8.1. Write a recursive function `MyGCD[a_,b_]`. How does its speed compare with Mathematica's `GCD` function?

The goal of this lab is to write our own PowerMod function. We have already written functions for taking quotients and remainders, so feel free to use Mathematica's `Quotient` and `Mod` commands.

Problem 8.2. Write a `PositivePowerMod[a_,b_,m_]` command that assumes that $b$ is positive and computes $\text{REM}\ (a^b, m)$. Remember to use recursion.

Problem 8.3. Use the extended Euclidean algorithm to find the inverse of 53 mod 137. Write it out very carefully so you can tell Mathematica how to do it below. What numbers do you need to keep track of as you work through the algorithm?

PROBLEM 8.4. Notice that to reconstruct the $s$ and $t$ values, you only need to know the quotients produced by the Euclidean algorithm.

Look at the example you worked out above, and write down the details of how $s$ and $t$ change.

PROBLEM 8.5. Fill in the rest of `GCDQuotients[a_,b_]`:

```
GCDQuotients[a_,b_]:=Module[{currentA=Abs[a], currentB=Abs[b], quotientsList={}},
   While[ b!=0,






   ];
   Return[quotientsList]
]
```

PROBLEM 8.6. Fill in the rest of `ListToST[quotientsList_]`. Hint: you can use `Last` to get the last element of a list and `Delete[ nameOfList, -1]` to get rid of the last element of a list.

```
ListToST[quotientsList_]:=Module[{currentS=  , currentT= },
   While[ Length[        ]      ,















   ];
   Return[{currentS,currentT}]
]
```

PROBLEM 8.7. Write a `ModularInversion[a_, m_]` command that combines `GCDQuotients` and `ListToST` to find an inverse to $a$ mod $m$.

PROBLEM 8.8. Write a `MyPowerMod[a_,b_,m_]` function that computes REM $(a^b, m)$ for any integer $a$ by combining `ModularInversion` and `PositivePowerMod`.

PROBLEM 8.9. How does `MyPowerMod` compare in speed and functionality to `PowerMod`?