Homework 3

Functions

The following problem set is worth 100 points. Please submit a zip file containing all source code I need to run your assignments using **QtSPIM**. Code will be graded on elegance and correctness.

Problem 1: Matrix Program (25pts)

Two matrices can be multiplied, provided that the number of columns on the left of the multiplication sign equals the number of rows on the right side of the multiplication sign. If the left matrix, call it **L** has m rows and n columns, and the right matrix, called **P** has n rows and p columns, then the resulting matrix, call it X has m rows and p columns with each element defined as:

$$X_{i,j} = \sum_{k=1}^{n} (R_{i,k} \; * \; L_{k,j})$$

Write a method called **mult** that takes addresses of two multi-dimensional arrays in the data segment and returns the address of a multiplication of the two matrices (also stored in the data segment). In your main method, print out the result of the method. Hard-code test data in the two addresses for testing purposes (stored as either row-major or column-major). If the multiplication cannot be performed, the method should return a null address and the main method should print an appropriate message with the result. Store your solution in **matrix.asm**.

Problem 2: Prime Problem (25pts)

A prime number is any number that is divisible by itself and 1. For example, 5 is a prime number since it can only evenly be divisible by 1 and 5. The number 6 is not prime since it can be divided by 1, 2, 3 and 6.

Write a function named **isPrime** that returns true if the argument input is a prime number and false if otherwise. Test your function in your main method with a couple examples. This solution should be named **prime.asm**.

Problem 3: Ackermann Problem (25pts)

Ackermann's Function is a recursive mathematical algorithm that can be used to test how well a system optimized its recursion performance. Design a function (with arguments m and n) which solves Ackermann's function. Use the following logic in your function:

- If m=0 return n+1
- If n=0 return Ackermann(m-1,1)
- Otherwise return Ackermann(m-1, Ackermann(m, n-1))

Once you tested your function, test it in your main with small values of m and n. Save your solution in a file named **ackermann.asm**.

Problem 4: Combination Problem (25pts)

The formula for computing the number of ways of choosing **r** different things from a set of **n** things is the following:

$$C(n,r) = n!/(r! * (n-r)!)$$

The factorial method **n!** is defined by:

$$n! = n * (n-1) * (n-2) \dots * 1$$

Discover a recursive version of the formula for **C(n, r)** and write a recursive method that computes the value of the formula. Call this method a few times in your main method. Save your solution in a file named **combination.asm**.

Submission Requirements

Please submit a zip file containing all source code I need for testing before the closing of the Canvas drop box.