Assignment 5

COMP 302 Programming Languages and Paradigms
Prakash Panangaden

Due Date: 4th April 2017

This assignment has two programming questions. Put the programming assignment in a file with the extension .fs as usual. There are two other questions that are not for credit; they are to help you study for the final.

Question 1[50 points] In this question you will implement a parser for the language of algebraic expressions described in class on the 9th of March. We use the following type definition to capture expression trees:

```
type exptree = Var of char | Expr of char * exptree * exptree
```

The input will be just plain strings and the output has to be an exptree. The input strings consist of simple algebraic expressions with `+` and `*` and **one-character** symbols that are just lower-case letters. It is also possible to use parentheses in the input expressions. Blanks are not allowed in the input[1]. The parser itself will consist of three mutually recursive functions as explained in class. There is a template on the course web site with more details and several examples. I am interested in making sure that the parser works correctly on valid input strings, I won't worry about dealing with bad strings and implementing error handling. I have not written the parser to detect all the possible problems. In the file called examples.fs you will see that not all bad strings are reported as bad strings, one simply gets the wrong parse tree.

Question 2[50 points] In this question you will implement a toy code generator for the very simple stack machine as explained in class on March 10th. Use the output from the parser as input to the code generator. The code generator should just print the "machine instructions" on the screen. Use statements like

```
printfn "LOAD  %c" c
printfn "ADD  %c" c
printfn "MUL  %c" c
printfn "STORE %i" tempstore
printfn "ADD %i" tempstore
```

[1] If you want to do something fancier please do so in your own files **but do not submit it!**

within your program to get this effect. There are examples on the web page. **PTO**

Question 3[0 points] Please derive a principal type of the following function using the sort of informal reasoning I have shown in class:

```
let S = fun x -> (fun y -> (fun z -> (x z) (y z)))
```

Of course you can type it in to the interpreter and see what the type should be. I am interested in your justification for why this is the type.

Question 4[0 points] Suppose we have a programming language called G-flat in which the type *int* is a subtype of $float$. Explain why $int \rightarrow int$ is not a subtype of $float \rightarrow float$. Please do not ramble on and on. The answer should be no more than 5 lines.