Homework #4

For those of the following questions requiring text or diagrams, attach a PDF file that contains all of the answers, clearly specifying the question that you're answering at each time.

Question 1 (5 pt.)

Consider the following regular expression matching positive or negative decimal numbers with an optional fraction:

```
[+-]? [0-9]+ (. [0-9]+ )?
```

a) Transform the regular expression to another regular expression that uses only concatenation, union, and Kleene closure operations.

b) Construct an NFA using the McNaughton-Yamada-Thompson algorithm that accepts the same language as the resulting regular expression, and provide its transition graph. Do not skip any steps of the algorithm, even if they insert redundant states. You can use multiple labels (or label ranges) on state transitions associated with character classes.

c) Write a C++ main program that takes a string as an argument and prints a message indicating whether the string matches the regular expression above. You can use class `NFA` by including file `NFA.h` from your main program. Attach only the main program in a file named `q1.cc`. This file should compile without modifications, and run correctly if linked together with file `NFA.cc`.

Question 2 (5 pt.)

Consider the following code:

```
My_2nd_var = 2;
if (x)
	return a == 1;                 ( ← TAB character before “return”)
```

a) Give the transition graph for the NFA of a scanner that recognizes all tokens present in this code. Remember to associate a label to each final state indicating which token it recognizes.

b) Store this code in a file named `input.txt`. Write a C++ main program that opens this file with an `std::ifstream` object, passes it directly to the constructor of `InputBuffer`, and scans its content with the NFA obtained before. The output of the program should be a list with the identifier of each token, followed by the associated lexeme. Attach only the main program in a file named `q2.cc`, which should compile without modifications when linked together with `InputBuffer.cc` and `NFA.cc`.