

Exercise 1 - Matching Parentheses

An important part of parsing most programming languages is to find enclosing scopes by matching the same parentheses. If parentheses are mismatched, then there is some sort of syntactic error in the program.

In the Stack lab, you will have to implement a small language translator that handles parentheses in the context of arithmetic expressions. But, this is a generally applicable problem that is easily solved with stacks.

This exercise challenges you to think of how to use a Stack to do simple parentheses (and brackets matching) using the following pairs of characters: (and), [and], and { and }.

Valid inputs should be balanced expressions like:

() [] {} ([]) {} ({})

Invalid inputs might be of the form:

({}) ((([]

Write some psuedo code or Java code that takes a string containing these characters and determines if it is balanced.

```
while input is non-empty
  char c = read character
  switch(c) {
    case '(':
    case '[':
    case '{': push c;
              break
    case ')': char d = pop;
              if (d != '(')
                throw mismatch exception
              break
    case ']': char d = pop;
              if (d != '[')
                throw mismatch exception
              break
    case '}': char d = pop;
              if (d != '{')
                throw mismatch exception
              break
  }

if stack is not empty
  throw mismatch exception
```