Email address problem

Email address checker

Algorithm : Email_add_check

Input(pre-condition): An arbitrary string

Output(Post-condition) : valid according to the specification or invalid with reasons(Error message)

Steps:

1. input <- an arbitrary string
2. format input
    - 2.1 . input.replace(“_at_”, “@”)
    - 2.2. input.replace(“_dot_”, “ . ”)
    - 2.3 . input.toLowerCase()
3. error= fxn_CheckAddress(input)
4. If error==0
    - print “input”

   Else
    - print fxn_error_Message(error);

• • • • •

Algorithm : error_message

Input(pre-condition): error as an int

Output(Post-condition) : a string indicating type of error

Step :

1. input <- error
2. if (input == 1) return "No @ symbol";
   if (input == 2) return "More than one @ symbol";
   if (input == 3) return "Invalid separator placement";
   if (input == 4) return "Contains invalid character";
   if (input == 5) return "Domain contains invalid character";
   if (input == 6) return "Missing square bracket(s)";
   if (input == 7) return "Brackets only allowed when numeric address";
   if (input == 8) return "Invalid numerical address";
   if (input == 9) return "Invalid extension";
   return "Invalid for some reason";

...

Algorithm : CheckAddress

Input(pre-condition) : a formatted String input

Output(Post-condition) : an int as Error

Steps:

1. address <- input
2. If (address.indexOf('@') == -1) return 1 // no @ symbol
3. if (address.lastIndexOf('@') != address.indexOf('@')) return 2; // two @ symbol
4. domain = address.substring(address.indexOf('@')+1) ; // Domain substring
5. for(i <- 0 to address.Length())
    1. char c = address.charAt(i)

    2. if (Character.isDigit(c)) {
        seensep = false;
        continue;
    }

    3. if (Character.isLetter(c)) {
        seensep = false;
        continue;
    }

CheckAddress continued…

4. if ((c == '[') || (c == ']')) continue;
5. if ((c == '@') || (c == '.') || (c == '-') || (c == '_')) {
    -if (seensep) return 3;
    -seensep = true;
    -continue;
   }
6. return 4;

6. numeric = true; bracketed = false;

7. for(i <- 0 to domain.length)

1. char c = domain.charAt(i)
2. if (Character.isDigit(c)) continue;
3. if (Character.isLetter(c)) {
    numeric= false;
    continue;
   }

. . . . . .

```
4. if (c == '.') continue;
                if ((c == '[') && (i == 0)) {
                        bracketed = true;
                        continue;
                }
5. if ((c == ']') && (i == (domain.length() - 1))) {
                        bracketed = true;
                        continue;
                }
6. return 5;
8. if (numeric) {
        if (domain.charAt(0) != '[')     return 6;
        if (domain.charAt(domain.length()-1) != ']') return 6;
    }
9. if (bracketed && !numeric) return 7;
```

...

```
10. if  (numeric) {
        1. domain = domain.substring(1, domain.length()-1);
        2. domain = domain.replace(".", " ");
        3. String[] parts = domain.split(" ");
        4.  try {
                for (int i = 0; i < parts.length; i++) {
                int j = Integer.parseInt(parts[i]);
                if ((j < 0) || (j > 255)) return 8;
                }
            } catch (Exception e) {
                return 8;
            }
        5. return 0;
    else
```

.....

```
    else{
                        extension = domain.substring(domain.length() - 4);
                            if (extension.equals(".com")) return 0;
                       extension = domain.substring(domain.length() - 7);

                            if (extension.equals(".com.au")) return 0;
                      extension = extension.substring(1);
                            if (extension.equals(".co.nz")) return 0;
                            if (extension.equals(".co.ca")) return 0;
                            if (extension.equals(".co.us")) return 0;
                            if (extension.equals(".co.uk")) return 0;
      } // end of else
    return  9;
```