

# Programming Problems

## 19<sup>th</sup> Annual Computer Science Programming Contest

Department of Mathematics and Computer Science  
Western Carolina University  
April 8, 2008

### Problem 1: Palindromes

Your program is to read a single line of user input and tell if yes or no it is a palindrome. If user inputs "quit" then exit the program.

A palindrome is a string that read from left to right or right to left looks the same.

For instance "abcba" is identical whether we read it from left to right or from right to left. Your program is expected to be case-sensitive.

On top of this definition we will consider the following pairs of characters to be *reversible*:

( ), [ ], { } , < >

This means that when reading from left to right we encounter the character '(' we should encounter instead the character ')' when reading from right to left.

For instance the string "ab(c)ba" will be considered to be a palindrome while "ab(cba" will not.

Here are some examples of palindromes:      aabbcbbaa    a

aa

a{a ( ) (a) ver{(aga)}rev

and some examples of non-palindromes: ab

( a(b(a

a(a a{b>c

Example of what your program output should look like:

```
>> input phrase (type 'quit' to exit the program):
```

```
aabbcbbaa
```

This is not a palindrome.

>> input phrase (type 'quit' to exit the program):

**(a)**

This is a palindrome.

>> input phrase (type 'quit' to exit the program):

**quit**

## Problem 2: Passwords

Your program will read user input and return a numeric value associated with that input. If user inputs "quit" then exit the program.

The user input is to be interpreted as a potential password.

The numeric value that you will return will be computed using the set of rules below. That value indicates the 'strength' of the password. The bigger the number, the more secure the password is considered to be.

Rules to compute the strength of a password:

- 0) By default the strength is 0. Use the rules below to find the proper value associated with your password.
- 1) If there are at least 8 characters, add +10 to the strength.
- 2) If longer than 8 characters, add +1 per every two-characters.
- 3) If the password contains both upper and lower case, add +1 to the strength.
- 4) If the password contains at least one numeric character, add +1 to the strength.
- 5) If the password contains at least one special character, add +1 to the strength.
- 6) For every two consecutive characters that aren't of the same type (alphabetic, numeric, special), add +1 to the strength.

Note:

- \* Alphabetic characters can be either lowercase (a..z) or uppercase (A..Z).
- \* Special characters are neither alphabetic or numeric.

Example of what your program output should look like:

```
>> password (type 'quit' to exit the program): abc
strength = 0
>> password (type 'quit' to exit the program): abcdefgh
strength = 10
>> password (type 'quit' to exit the program): abcdefghi
strength = 10
>> password (type 'quit' to exit the program): abcdefghij
strength = 11
>> password (type 'quit' to exit the program): abcdefghij0
strength = 13
```

```
>> password (type 'quit' to exit the program): Abcdefghij0  
strength = 14  
>> password (type 'quit' to exit the program): 0123456789  
strength = 12  
>> password (type 'quit' to exit the program): 66Nk+79=abc  
strength = 19  
>> password (type 'quit' to exit the program): quit
```

### Problem 3: Parentheses Check

Your program will read user input and return a message indicating if the input was valid or not. If user inputs "quit" then exit the program.

The user input is to be interpreted as a single line of text.

You will parse this string of text and indicate whether parentheses, square brackets, and curly braces are properly opened and closed. Any input that is not a parenthesis/bracket/brace can be ignored.

For instance here a few things you may want to consider:

- \* Is there a closing parenthesis/bracket/brace for every opening one?
- \* Are the parenthesis/bracket/brace blocks properly embedded one into another?
- \* anything that is not a parenthesis/bracket/brace can be ignored.

Examples of invalid input:

<code>((is very )</code>	is invalid because it misses a closing parenthesis
<code>(is very) )</code> parenthesis	is invalid because there is an extra closing parenthesis
<code>(aaa [bbb ) ccc]ddd</code>	is invalid because it closes the parentheses before the brackets

Example of valid input:

`apples (are very [very] good {at} (k(ee)[p])ing ) the [doctor{away}]`.

Example of what your program output should look like:

```
>> user input (type 'quit' to exit the program): ((is very )
Invalid.
>> user input (type 'quit' to exit the program): (is very) )
Invalid.
>> user input (type 'quit' to exit the program): ((is very)
)
Valid.
>> user input (type 'quit' to exit the program): quit
```

## Problem 4: Rock Paper Scissors

Your program will read user input and use it to play a round of the well-known rock/paper/scissors game. If user inputs "quit" then exit the program.

The user input is case insensitive and the user is expected to type "paper", "rock", or "scissors". If anything else is typed then call for another round giving an invalid input error message.

At each turn, your program is to pick arbitrarily and unsystematically one of "paper", "rock", "scissors." You shouldn't use user input in deciding what to play.

After your program is done picking what to play, compare the user input and the computer's choice, and indicate who won that round. If there is a draw say so.

Here is a table describing which choice wins over which other choices:

<u>usr \ cpu</u>	<u>Paper</u>	<u>Rock</u>	<u>Scissors</u>
<b>Paper</b>	draw	usr wins	cpu wins
<b>Rock</b>	cpu wins	draw	usr wins
<b>Scissors</b>	usr wins	cpu wins	draw

Example of what your program output should look like:

```
Please type rock, paper, or scissors (type 'quit' to exit the program).
```

```
>> SCISSORS
```

```
Computer Choice:2
```

```
The computer won (rock > scissors).
```

```
Please type rock, paper, or scissors (type 'quit' to exit the program).
```

```
>> rock
```

```
Computer Choice:2
```

```
This round is a draw.
```

```
Please type rock, paper, or scissors (type 'quit' to exit the program).
```

```
>> paper
```

```
Computer Choice:2
```

```
You won (paper > rock).
```

```
Please type rock, paper, or scissors (type 'quit' to exit the program).
```

```
>> quit
```