Written Homework #3
CS 162: Introduction to Computer Science

Alves Silva, Otavio Augusto
PSU ID: 902840168

1. Create an algorithm for outputting the multiplication table

1. Create the variables.
    a. line
    b. multiplier
    c. result
2. Put the initial values of each variable.
    a. line = 1
    b. multiplier = 1
    c. result = 0
3. Create two loops
    a. One for the amount of lines.
        i. Increase by one the variable line.
        ii. Replace the value of multiplier for one.
        iii. Output a newline in the prompt.
        iv. The loop will continue until line is equal or less than the number 5.
    b. One for number of times that the variable line will be multiplied
        i. Calculate the amount of variable result
            1. result = multiplier * line
        ii. Output value of variable result in the prompt
        iii. Increase by one the variable multiplier.
        iv. The loop will continue until the variable multiplier is equal or less than the number 10.
4. Put the loop for the multiplication inside the loop of lines.

2. Rewrite one of the examples in Chapter 5

A code example with a break keyword.

```
sum = 0;
cin >> num;
while (cin)
{
          if (num < 0)
          {
            //if num is negative, terminate the loop
            cout << "Negative number found in the data." << endl;
            break;
          }
          sum = sum + num;
          cin >> num;
}
```

The same code example, but rewritten without a break keyword.

```
sum = 0;
cin >> num;
//if num is negative, terminate the loop
while (num >= 0)
{
        sum = sum + num;
        cin >> num;
}
cout << "Negative number found in the data." << endl;
```

3. Create an algorithm for the testing software

1. Check the function *intro()*
    a. Are the messages being printed correctly?
    b. Are the rules and information correct and clear?
2. Check the function *read_phrase()*
    a. Is the function reading the phrase?
        i. Output this values only one time to check.
    b. If the user entered more than the size of the phrase an error is printed?
    c. Does the function have a loop to continue asking for a valid phrase?
3. Check the function *read_word()*
    a. Is the function reading the word?
        i. Output this values only one time to check.
    b. If the user entered more than the size of the word an error is printed?
    c. Does the function have a loop to continue asking for a valid word?
4. Check the function *uppercase()*
    a. Does the function uppercase each character in phrase and word?
        i. Output these values one time to check.
5. Check the function *hidden_word()*
    a. Is the word in the phrase?
        i. If not, do loop and ask again until the user types a valid word.
6. Check the function *guessing_word()*
    a. Is the function reading the guessing word?
        i. Output this values only one time to check.
    b. Does the function uppercase each character in the guessing word?
        i. Output this values only one time to check.
    c. Is the function comparing the hidden word with the guessing word?
        i. Output the comparison value one time to check.
    d. Is the function calculating the player points and remain chances?
        i. Output the values each time that a player does a mistake.
    e. Is the function returning the player points?
7. Check the function *results()*
    a. Is the function comparing the right values?
    b. Is the function outputting the winner and player points?

4. Style

The code has all functions, variables, arguments and conditions with clear comments that show their purpose and application in code. I could write a function to do turn of each player. Therefore, this function would have all previous functions except the *intro()* and *results()*. It would help my code has a clear function *main()*, as would have less lines. The code algorithm has more than seven hundred words, which are explaining each step in line of processing, helping an user or programmer that wants to know how the program works.

References

- MALIK, D.S., C++ Programming: From Problem Analysis to Program Design, Sixth Edition, 2013.