Written Homework #2
CS 162: Introduction to Computer Science

Alves Silva, Otavio Augusto
PSU ID: 902840168

Create an Algorithm for the process of checking your *pdx* Gmail.

1. Turn on your computer.
2. After the computer initialization, you will need to join in a network with access to Internet.
3. After established connection, please open your default browser. For example, *Google Chrome* or *Mozilla Firefox*.
4. Enter the follow address in the address bar in your browser and press the button *Enter*.
    a. mail.pdx.edu
5. If you have any HTTP problem. Check your Internet connection or choose another network.
6. If everything happened without problems. You will see a page asking your pdx email and password. Enter the following information as below.
    a. Login: <Odin Login>@pdx.email – Without the " < > "
    b. Password: Odin password
7. If you have any problem. Check your Odin account at the following link.
    a. banweb.pdx.edu/
8. If everything happened without problems. You already have access for your *pdx* Gmail and enjoy the services provided.
    a. You have access for you inbox, sent email, drafts and more.
    b. You can compose an Email.
    c. You can see your trash and spam folders.

Terminology

- **Loop invariant**

    To explain the meaning of loop invariant, we need to know the meaning of loop and invariant in this context. Loop is a sequence of instructions and statements that are continually repeated until a certain condition. Invariant in this context, is a property that remains true every time the loop body is executed. Therefore, a loop invariant is defined by a property that is true before and after the loop.

    A simple example of loop invariant can be showed using the keyword *while*. The following part of a code shows how it can be used.

```
int i=0;
n=10;

// Is "i" less or equal than "n"? Yes.
while (i <= n)
{
  cout << i << endl;
  ++i;
}
```

After the loop, the *while* condition continues true. Therefore, it's a loop invariant.

- **Side effect**

Side effects are modifications made by a function or expression, which modify states in the computer, like a variable value or the action of write data in the disk, or which have some interaction with the outside world, like the action of disable a button in a GUI (Graphic User Interface).

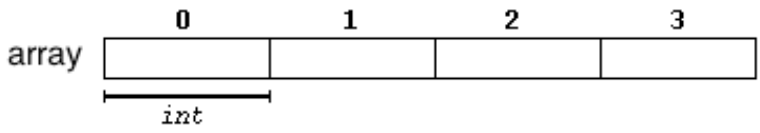
An example of side effect can be seen by the following code, which has the side effect of writing data to output. The function *output* doesn't modify the value of the variable, but it has effect in the "outside world".

```
char output(char data[])
{
     return cout << data << endl;
}

int main()
{
    char initial[2];
    cout << "Enter your first initial: ";
    cin >> initial;
    output(initial);

    return 0;
}
```

- **Arrays in C++**

An array is a collection of elements that have the same data type and are placed systematically in the contiguous memory locations. It can be individually referenced using an index.

Therefore, create more than just one value of some data type is easier when the programmer uses an array than create one by one. For example, an array of integers of four values can be represented as:



Each black space represents an element of the array. In this example, the data type is integer. The number zero represents the first element. The declaration of an array in a C++ code is:

data_type variable_name[number_of_elements];

Therefore, if we declared the same array like above. It will be like the following line:

```
char array[4];
```

Testing

Test the code is an important step towards the creation of good software that user can trust and use with the minimum occurrence of errors and problems. The forms of test are normally divided in two categories, which are static testing and dynamic testing.

- **Static testing**

It is a form to test the code without using him. Actually, it isn't a detailed testing, but checks the readability, maintainability, algorithm and coherence of the code. Normally, the developer who wrote the code does it with an eye toward completeness or appropriateness for the task at hand. This form of testing is a good way to reduce the costs fixing problems that would happen in the develop cycle. The static method of testing involves verification, which cares with the design and code. These are some static testing methods examples:

Inspection: It is a form of software peer review by a group of specialized individuals using a defined process that looks for software requirements specifications and test plans.

Walkthrough: It is another form of software peer review that is done by group of specialized individuals, but they go through asking questions, making comments about possible errors, finding violation of development standards, and other problems.

- **Dynamic testing**

It is a form to test the code dynamically. Therefore, this form uses him at the same moment that is tested. The analysis of the code is done examining the physical responses of the system with variables that aren't constant and change with time. For this to be done, it must use input values and check if the output is what was expected. In other words, we need compile, run the program and input values to check the results. This form of testing is a great away to see exactly the error by passing the development time. The dynamic method of testing involves validation, which cares with the testing the real purpose and objective of the code. An example of dynamic testing is the "unit testing".

Unit testing: It is a dynamic software testing method where some individual parts of the main code are tested to determine whether they are fit for use. The best benefit of this method is possibility to see if each individual part is correct. Therefore, the programmer can find the problems early, change and integrate the code easily, and have a good design and documentation.

Ethics and Security

The risk of identify theft could be limited using some of these examples when the software designer is building the program:

1. The user privacy should be a top priority at the time this developing the software because in some cases the user's life can be affected dramatically.
2. Encryption must be used to protect the access of sensitive information.
3. Authentication of users could be used to have sure about the person who is using the program or trying to access some information.
4. Granular access control must be built to not show everything to the user.
5. The program, which storages sensitive data, must have secure and protected systems for its databases and servers.

References

- MALIK, D.S., C++ Programming: From Problem Analysis to Program Design, Sixth Edition, 2013.
- Loop invariants
  - http://www.cs.uofs.edu/~mccloske/courses/cmps144/invariants_lec.html
- Side effect
  - http://programmers.stackexchange.com/questions/40297/what-is-a-side-effect
- Arrays
  - http://www.cplusplus.com/doc/tutorial/arrays/
- Software testing
  - http://en.wikipedia.org/wiki/Software_testing#Testing_methods
- Data security
  - http://docs.oracle.com/cd/B10501_01/network.920/a96582/overview.htm#1004641