PIC 10A 1C. Week 7b Exercises. TA: Eric Kim. [Solutions] *(v3 Updated: 1/21/2016)*

1. AT-AT

Write a program that, given a user-given sentence, replaces all 'a' with '@':

```
    Enter a sentence: Dan ate apples, haha!
    D@n @te @pples, h@h@!
```

Rule: Only use a single **range for-loop** to iterate over the input string.

[Solution]

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    cout << "Enter a sentence: ";
    string s;
    getline(cin, s);
    for (char c : s) {
        if (c == 'a')
            cout << '@';
        else
            cout << c;
    }
    return 0;
}
```

2. Don't Britta This!

Consider the following class interfaces defined in `person.h` and `studyroom.h`:

```
class Person {
private:
    string myname; unsigned int myage;
public:
    Person(string name, unsigned int age);
    unsigned int get_age();
    string get_name();
};
class StudyRoom {
private:
    vector<Person> myv; // Stores People
public:
    StudyRoom();
    void add_person(Person p); // Add person to room
    double compute_avg_age(); // Avg age of people in study room
};
```

Write a program that does the following:

1. Ask the user for a name and an age.
2. Create a Person object, and add it to the StudyRoom.
3. Ask the user if he/she is finished. If not, then go back to 1.
4. Else, output the average age of the people in the StudyRoom (in fixed precision, with 2 decimal points).

Example:

```
    Name? Jeff Winger
    Age? 34
    Done? y/n: n
    Name? Britta Perry
    Age? 30
    Done? y/n: n
    Name? Troy Barnes
    Age? 21
    Done? y/n: y
    Avg age: 28.33
```

Rule: Only use a single **do-while** loop.

[Solution]

```
#include <iostream>
#include <iomanip>
#include <string>
#include "person.h"
#include "studyroom.h"
using namespace std;
int main() {
    string name; unsigned int age; string resp;
    StudyRoom studyroom;
    do {
        cout << "Name? ";
        getline(cin, name);
        cout << "Age? ";
        cin >> age;
        Person p(name,age);
        studyroom.add_person(p);
        cout << "Done? y/n: ";
        cin >> resp;
        cin.ignore();  // Important! Mixing getline() and cin.
    } while (resp == "n");
    cout << "Avg age: " << fixed << setprecision(2) <<
studyroom.compute_avg_age();
    return 0;
}
```

3. The Overzealous Censor Officer

Write a program that replaces every other character in a string with '*'. Example:

```
    Enter a string: Frankly my dear
    F*a*k*y*m* *e*r
```

Rule: Only use a single **for loop**.

[Solution]

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    cout << "Enter a string: ";
    string s;
    getline(cin, s);
    for (size_t i = 1; i < s.size(); i+=2) {
        s[i] = '*';
    }
    cout << s;
    return 0;
}
```

4. Blackjack-Lite

Write the following program:

1. Ask the user for an integer, and add it to a running sum.
2. If the current sum is greater than 21, output "Bust!" and exit.
3. Else, if the current sum is equal to 21, output "Blackjack!", and exit.
4. Else, go back to (1).

Example:

```
[You: 0] Integer? 7
[You: 7] Integer? 3
[You: 10] Integer? 11
Blackjack!
```

```
[You: 0] Integer? 6
[You: 6] Integer? 9
[You: 15] Integer? 8
Bust!
```

Rule: Only use a single **while** loop - no other looping constructs allowed.

[Solution]

```
#include <iostream>
using namespace std;
int main() {
    unsigned int sum = 0, n;
    while (sum < 21) {
        cout << "[You: " << sum << "] Integer? ";
        cin >> n;
        sum += n;
    }
    if (sum == 21) {
        cout << Blackjack!";
    } else if (sum > 21) {
        cout << "Bust!";
    }
    return 0;
}
```

5. #justpic10Athings

We would like to write a program that, given a string of '+' and '#', computes a more compact version of the original string. For instance, here are some expected outputs:

`"+++##"`
`Becomes: "+3#2"`
	`"++++#++"`
	`Becomes: "+4#1+2"`

`"+#"`
`Becomes: "+1#1"`
	`"+++++"`
	`Becomes: "+5"`

`""`
`Becomes: ""`
	`"#++###++++"`
	`Becomes: "#1+2#3+4"`

Write a program that, given such a user-inputted string, outputs its compressed version:

```
    Enter a string: ++###
    +2#3
```

Aside: This is a form of run-length encoding, a technique used to compress data into a smaller (yet equivalent) form. For instance, when you compress a file to a .zip file, the compression program is likely using this principle to achieve a much smaller file size!
As you can imagine, some types of data are more amenable to compression that others. A file with lots of long runs, ie "+++++++++", compress well, whereas files with only short runs, ie "+#+#+#+", compress poorly.

[Solution]

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    cout << "Enter a string: ";
    string s;
    getline(cin,s);
    if (s.size() == 0)
        return 0; // Terminate early.
    size_t i = 0, cnt = 0;
    bool isplus = (s[0] == '+'); // initialize
    while (i < s.size()) {
        if (isplus && (s[i] == '+')) {
            cnt = cnt + 1;
        } else if (isplus && (s[i] == '#')) {
            /* Switch! '+' -> '#' */
            isplus = false;
            cout << "+" << cnt;
            cnt = 1; // We've seen 1 '#'
        } else if (!isplus && (s[i] == '#')) {
            cnt = cnt + 1;
        } else if (!isplus && (s[i] == '+')) {
            /* Switch! '#' -> '+' */
            isplus = true;
            cout << "#" << cnt;
            cnt = 1; // We've seen 1 '+'
        }
        i = i + 1;
    }
    /* Output final run */
    if (isplus)
        cout << "+" << cnt;
    else
        cout << "#" << cnt;
    return 0;
}
```

Here's an alternate solution (suggested by a student) where each iteration of the for-loop explicitly looks ahead to determine if we are switching from '+' to '#' (or vice-versa). To avoid indexing out of bounds, we add a padding character 'X' to the end of the input string, and take care to only iterate from 0 to (s.size()-1). In my opinion, this code is easier to understand:

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    /* Ask user for input string */
    cout << "Enter a string: ";
    cin >> s;
    if (s.size() == 0)
        return 0; // if input is empty, program terminates early
    /* plus, hash keep track of nb of '+','#' seen during current run */
    unsigned int plus = 0, hash = 0;
    /* Hack: add padding char 'X' at end to avoid indexing s out of bounds */
    string spad = s + "X";
    /* Perform runlength encoding of s */
    for (size_t i = 0; i < (spad.size()-1); ++i) {
        if (spad[i] == '+') {
            ++plus;
            if (spad[i + 1] != '+') {
                // prints out the + value if the following value is not a +
                cout << "+" << plus;
                plus = 0; // resets the plus value
            }
        } else if (spad[i] == '#') {
            ++hash;
            if (spad[i + 1] != '#') {
                // prints out the # value if the following value is not a #
                cout << "#" << hash;
                hash = 0; // resets the hash value
            }
        }
    }
    cin.ignore();
    cin.get();
    return 0;
}
```