Homework #1 v1.0 – A Password Cracker

Marin Šarić

Your rule list code is: `4TYCVRWC6YRAX`

Contents

> **Hint:** Reading through the whole assignment before attempting to solve it might save you some time

1 Read This First

- This is a programming assignment. Each student is getting a different assignment. You must solve this assignment on your own.
- You will need to write and submit a solution in either C++, Java or Python. You cannot mix languages in the assignment.
- You will be able to check your solution online on

  `http://130.237.218.85/~marin/hw1check/`

- You need to upload your solution to `BILDA`.
- Your submission must conform with the specifications set out in Section 5.
- Your assignment will be evaluated automatically. There will be no partial credit.
- You will need the *rule list code* from the beginning of this document.

2 Introduction

You will write a program that will try to crack a password based on some hints. The problem consists of parts `A` and `B`, each worth `5` points.

1. Each password character is going to be one of these `9` uppercase letters, from `A` to `I`:

   ```
   ABCDEFGHI
   ```

2. No other characters will appear in the password.

3. The password is N characters long.

If you knew nothing else about the password, the number of valid password combinations would be 9^N. However, if you know more about the password, you can eliminate some combinations.

Your program will "attempt" passwords by printing them out to screen.

3 Part A (5 points)

- Assume the password is `4` characters long.
- Without hints, you would have to try $9^4 = 6561$ passwords, everything from `AAAA`, `AAAB` all the way to `IIII`.
- Thankfully, you have additional information on these passwords:

```
- The last letter of the password is either 'E' or 'D'.
- Unless you are looking at the last letter of the password:
   If a letter 'C' occurs in the password,
    the following letter will be either 'I' or 'F'.
- The letter 'C' can occur at most 3 times in the password.
```

1. Write a program that prints out all of the combinations that are possible using the hints above.
2. Each password needs to be printed on a separate line. No other text should be printed.
3. The program should not wait for any input (no waiting to press ENTER, etc.)
4. Copy-paste the passwords into the online form to make sure your solution is correct.
5. Save your program as `solution_a.cc`, `SolutionA.java` or `solution_a.py`

4 Part B (5 points)

1. Modify your program from part A so that it works for N characters, where N is any number from `1` to `9`. The number N should be accepted as a command line parameter, see section 5 for details.

2. The program should not wait or ask for any input (waiting to press ENTER, typing N, etc). Number N is passed as a command line parameter.

3. Modify your program so that it does not print the passwords anymore, but just outputs the total number of passwords.

4. Copy-paste the total number of passwords for passwords that are $N = 7$ characters long into the online form to make sure your solution is correct.

5. Save your program as `solution_b.cc`, `SolutionB.java` or `solution_b.py`

5 Submit your solution

You need to submit a `zip` archive *usern*`_hw1.zip`, where *usern* is your KTH username (for example `johang_hw1.zip`).

The contents of the `zip` archive should be:

- `rulecode.txt` – This file should contain your *rule list code*, nothing else. You will find your rule list code on the first page of this document, near the document title.
- `email.txt` – This file should contain your KTH email, nothing else.
- `parta.txt` – This file should contain all the passwords for part A, nothing else.
- `partb.txt` – This file should contain a single number, the total number of 7 character passwords, nothing else.
- `solution_a.cc` or `SolutionA.java` or `solution_a.py` – The solution for part A.
- `solution_b.cc` or `SolutionB.java` or `solution_b.py` – The solution for part B.

Attention:
Do not put **any other files** in your `zip` file.
Do not put **any subfolders** in your `zip` file.
Use the online form to verify the `zip` file.

5.1 C++ notes

You can program your solution on Windows, MacOS X or Unix using standard C++. The last few versions of Visual Studio on Windows, XCode on MacOS and g++ on Linux will work just fine.

The only supported library is the standard C++ library. Your solution cannot depend on other `cc` files or third party libraries to run. Windows or MacOS specific libraries or C++ extensions are not allowed.

Here is a description of the system we will use to test your program: Our evaluation system will run Ubuntu Linux. We will use the `g++` compiler version `4.5` from the GCC compiler suite to compile your solution.

- The compilation commands will be:

```
g++ solution_a.cc -o solution_a
g++ solution_b.cc -o solution_b
```

- Part A will be executed as:

```
./solution_a
```

- Part B will be executed as:

```
./solution_b 7
```

5.2 Java notes

You can program your solution on Windows, MacOS X or Unix using a reasonably recent version of Java.

Your solution cannot depend on other `java` files or third party libraries to run. The only supported library is the standard Java library. Windows or MacOS specific libraries or Java extensions are not allowed.

Here is a description of the system we will use to test your program: Our evaluation system will run Ubuntu Linux. We will use the official Oracle's `javac` compiler for Java 6.

- The compilation commands will be:

```
javac SolutionA.java
javac SolutionB.java
```

- Part A will be executed as:

```
java SolutionA
```

- Part B will be executed as:

```
java SolutionB 7
```

5.2.1 Putting mutliple classes in one `java` file

You can put multiple classes in one `java` file by declaring only one of them `public` and omitting the `public` declaration from other classes. For example:

```
public class SolutionA {
    public static void main(String []args) {
        ...
        class_b = new HelperClass();
        ...
    }
```

```
    HelperClass class_b;
};

class HelperClass {
     ...
};
```

5.3 Python notes

You can program your solution on Windows, MacOS X or Unix using a resonably recent version of Python.

You cannot depend on other `py` files or libraries outside the standard Python distribution.

Here is a description of the system we will use to test your program: We will use Python version `2.7` to run your solution. Our evaluation system will run Ubuntu Linux.

- Part A will be executed as:

  ```
  python solution_a.py
  ```

- Part B will be executed as:

  ```
  python solution_b.py 7
  ```