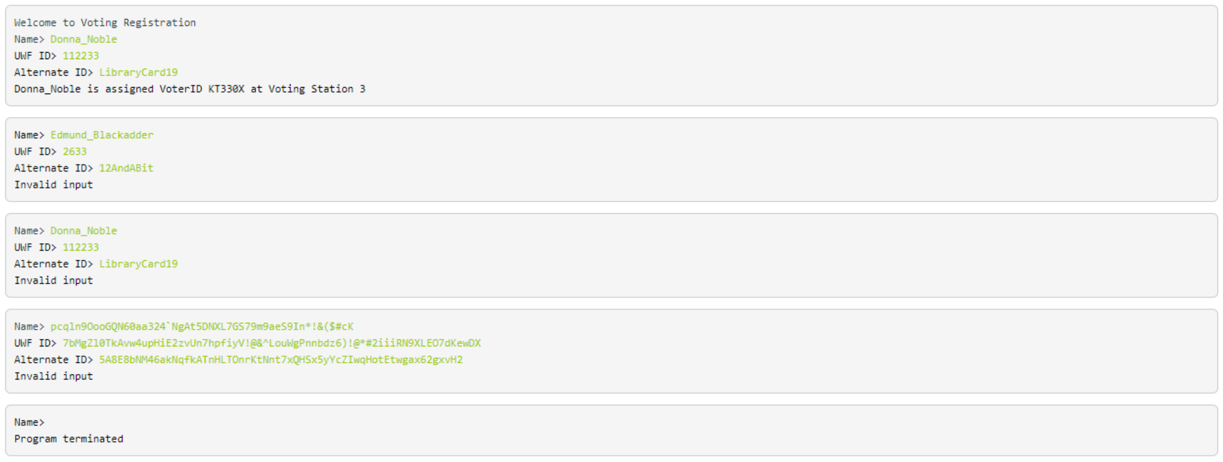
This is a sketch of the Code for the Voter Registration, it omits tons of security-related issues, such as data security, etc. Here we just concentrate on memory safety issues in C(++).

Asignment

In C/C++ write a program that has the following components:

1. It accepts a fixed text file in a format specified below. This file contains data describing a person by name (String), UWF ID (unsigned int) and acceptable alternate ID (String). The file name is **input.dat** (hard-coded is fine!).
2. It creates a sequential file that contains some records from the first file, but with an added Voter ID (String) and Voting Station number (unsigned int). The file name is **output.dat** (hard-coded is fine!).
3. It reads (repeatedly) input from stdin/cin (it's ok to have a prompt, but it's important that each input is on a separate line).
    1. A name as a String
    2. A UWF ID (as an unsigned int)
    3. An (hypothetical) alternate ID (a String)
4. It then scans the input file to check if this information is contained in the file.
5. If that is the case, it also makes sure (how it does that is up to you) that this information hasn't been entered before (in the same program run). Use the UWF ID to determine this - we just assume it's unique.
6. If either of these tests fails, a rejection message is printed - this message should **not** give any information about the cause of the failure!
7. If both tests succeed, the program assigns a **unique** Voter ID String and a Voting Station number (int between 1 and 9, in both cases I don't really care however you do that) and stores these values together with the inputs in the output file.
8. A confirmation is displayed, showing Voter ID and Voting Station number.
9. If the name is empty, the program terminates.
10. If any input is technically illegal (see below for a couple of ideas I'm going to use), it must not crash. Even if one input is illegal it still has to ask for the subsequent ones and reject without any additional information after the "Alternate ID" input.

Example run (user input in green)



```
Welcome to Voting Registration
Name> Donna_Noble
UWF ID> 112233
Alternate ID> LibraryCard19
Donna_Noble is assigned VoterID KT330X at Voting Station 3
```

```
Name> Edmund_Blackadder
UWF ID> 2633
Alternate ID> 12AndABit
Invalid input
```

```
Name> Donna_Noble
UWF ID> 112233
Alternate ID> LibraryCard19
Invalid input
```

```
Name> pcqln9OooGQN60aa324`NgAt5DNXL7GS79m9aeS9In*!&($#cK
UWF ID> 7bMgZl0TkAvw4upHiE2zvUn7hpfiyV!@&^LouWgPnnbdz6)!@*#2iiiRN9XLEO7dKewDX
Alternate ID> 5A8E8bNM46akNqfkATnHLTOnrKtNnt7xQHSx5yYcZIwqHotEtwgax62gxvH2
Invalid input
```

```
Name>
Program terminated
```

Input file format:

Per line: User name (String without any whitespace), UWFId (int without a sign), alternate ID (String without any whitespace), each separated with at least one whitespace)

Example: input.dat

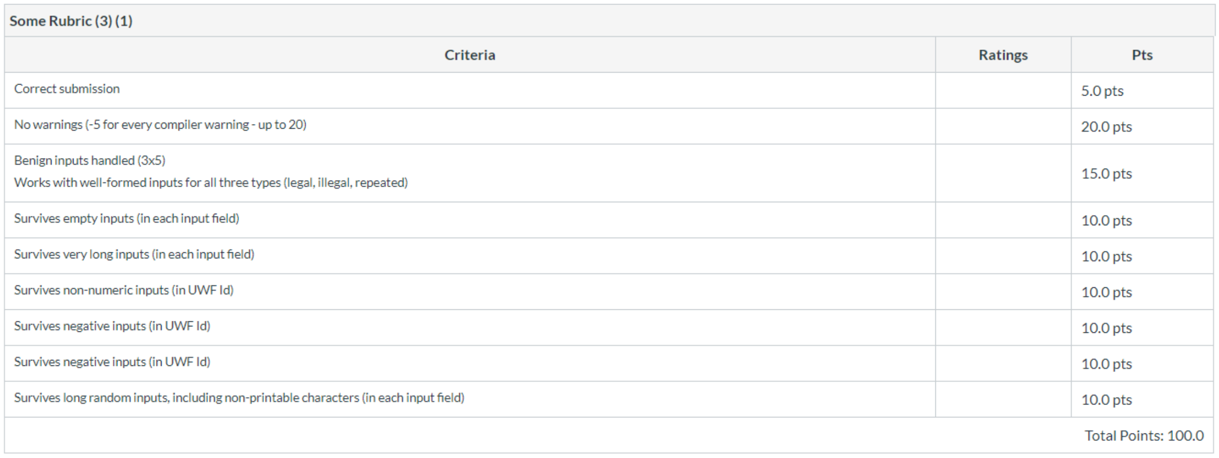
What am I going to do to your program?

Be prepared that I'm running each submission through a certain process **(everything will happen in my copy of the course VM)**

1. Compile it (see guidelines). Warnings will meet deductions
2. Run a legal case (should accept)
3. Run an illegal case (should reject)
4. Run a repeating case (should reject)
5. Run a variety of illegal inputs, including but not limited to
   1. no input - just ENTER (fine for name to end the program)
   2. Long inputs (500 characters+)
   3. Non-numeric inputs
   4. Numeric inputs that will overflow an unsigned int
   5. Negative inputs
   6. ...
6. I will also use another **input.dat**, but you can rely on the fact that it's built along the rules of your test file.
7. Or not...

Submission guidelines

1. Just drop the **main.c (main.cpp) (and** all other eventually necessary .c(pp) and .h files) into the dropbox. I will just place them in a directory on the virtual machine image. It is your responsibility to make sure they are complete and compile without warnings!
2. I will compile using **gcc -Wall -o a.out *.c**. or **gcc -Wall -o a.out *.cpp**



Some Rubric (3) (1)

Criteria	Ratings	Pts
Correct submission		5.0 pts
No warnings (-5 for every compiler warning - up to 20)		20.0 pts
Benign inputs handled (3x5) Works with well-formed inputs for all three types (legal, illegal, repeated)		15.0 pts
Survives empty inputs (in each input field)		10.0 pts
Survives very long inputs (in each input field)		10.0 pts
Survives non-numeric inputs (in UWF Id)		10.0 pts
Survives negative inputs (in UWF Id)		10.0 pts
Survives negative inputs (in UWF Id)		10.0 pts
Survives long random inputs, including non-printable characters (in each input field)		10.0 pts
		Total Points: 100.0