Hw01StudentID

This homework assignment is meant to make sure you can write, compile, run and **debug** simple C programs using IDE and its debugger. This is an individual assignment; you may not share code with other students. You will need to know how to compile and run C programs and how to debug them basically to have a jump start of this course.

This assignment provides an introduction to some of the major features of debugger (from any IDE) by leading you through the debugging process of a slightly buggy program. A debugger allows the programmer to stop a program in the middle of its execution so that he or she may observe how the program branches and executes. Debuggers can help speed the process of isolating and correcting bugs.

Homework01 - Due midnight Tuesday, March 11, 2014, 3 points

First of all, read the class hand-out "DS Jump Start and Debugging".

1. **Create a directory structure as recommended** in the class hand-out and store away your files into the corresponding folders. In a real world c/c++ programming development environment, it is very practical and beneficial that the source files are separately stored or managed depending on their types. Set up the folders such as **/lib, /include, /src** first. Then start IDE and place your source files accordingly. You may need to practice a few trials to understand how IDE create or handle their files properly.
2. Get a copy of the program **hw01Buggy.cpp** from Piazza. Add it into your IDE environment, and follow the instructions below.
3. Post your source file(**hw01StudentID**.cpp) and answer file(**hw01StudentID.txt**) in Piazza.

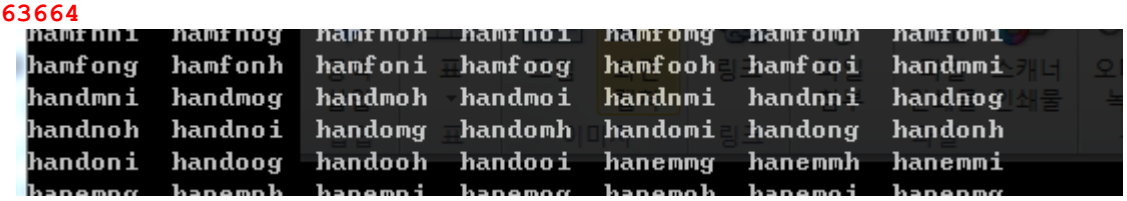
Background

A Mobile Telephone keypad

(The original program written for unix/gcc development environment in UC Berkley is modified for Windows and c programmers.) Follow the instructions outlined in the assignment in order to debug the provided program and answer the questions throughout the assignment on a separate piece of paper. You will be required to present **the corrected code and answers to the questions** in this assignment. The questions in this assignment are meant to be thought-provoking. They illustrate important aspects of the C language. If you experience problems, do not hesitate to post them on **Piazza** for help! If that does not work, ask TA's.

The provided program is to turn regular phone numbers into "words." For example, we may have a number 1-800-**426-3664** that translated into 1-800-**HANDONG**. (When you try **356-9377** with **five** consonants, you will find "**flowers**", **232-6459** with **four** consonants "**afamily**".) The program, when working properly, will provide all possible "words" that have no more than a user-specified number of consonants. Below is a transcript of how the program should work, with user input appearing in boldface:

```
This program finds words that can be associated with phone numbers.
What is the maximum number of consonants you would like in the generated
Word to possess?(note: 0's and 1's are considered consonants)
4
Please enter a 7 digit phone number with dashes removed
4263664
```

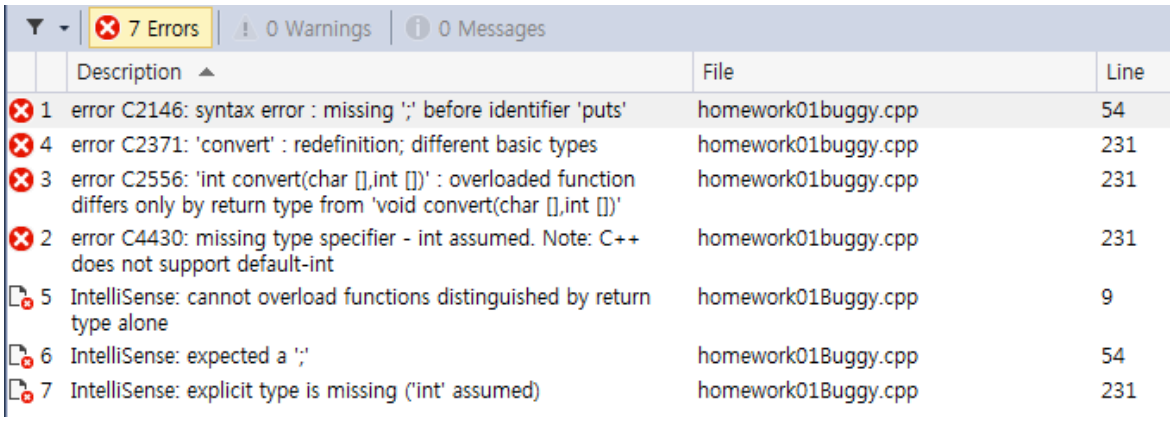



```
Would you like to try another number? (Y/N)
n
```

Before continuing on with this assignment, understand how the program is supposed to work.

Assignment

The first step in the debugging process is the compilation of the program. In the process of compiling code, useful error messages are provided that identify problematic parts of the code that are not syntactically correct. Very often, these mistakes are typos. You should see something similar to the following (your output may not be the same anyway…):



7 Errors | 0 Warnings | 0 Messages

	Description	File	Line
1	error C2146: syntax error : missing ';' before identifier 'puts'	homework01buggy.cpp	54
4	error C2371: 'convert' : redefinition; different basic types	homework01buggy.cpp	231
3	error C2556: 'int convert(char [],int [])' : overloaded function differs only by return type from 'void convert(char [],int [])'	homework01buggy.cpp	231
2	error C4430: missing type specifier - int assumed. Note: C++ does not support default-int	homework01buggy.cpp	231
5	IntelliSense: cannot overload functions distinguished by return type alone	homework01Buggy.cpp	9
6	IntelliSense: expected a ';'	homework01Buggy.cpp	54
7	IntelliSense: explicit type is missing ('int' assumed)	homework01Buggy.cpp	231

or

The numbers after the filename refer to the numbers of lines that contain errors. It is important to resolve the **first** error message (in red) first. "Parse errors" typically signify something that is syntactically incorrect. "Warnings" are issued for code that the C compiler believes to be problematic but that is syntactically correct. Warnings will not prevent your code from compiling, but they can tip you off to some rather serious problems.

1. Correct the errors. Make a note of what changes you made. You may need to make small changes and recompile to see if those errors are fixed. This may take several iterations. It is okay not to fix all the warnings but all of the non-warning problems will need to be fixed. Take care of the first error **first** since the first one may cause the following ones. If you reach this point:

   ```
   Error C4996: 'gets': This function or variable may be ……:
   ```

   You may follow one of their recommendations. It is a Microsoft related stuff. You may set set <**SDL checks**> to **No** in Proejct → Properties → C/C++ → General.
   Or add -D _CRT_SECURE_NO_WARNINGS in the compiler command line option.
   If all do not work, google it^^.

Let's see if the program works. To run it, select **DEBUG → Start without debugging**.

```
This program finds words associated with phone numbers.
What is the maximum number of consonants you would like to have in the
generated word to possess? (note: 0's and 1's are considered consonants)
4
Please enter a 7 digit phone number with dashes removed
4263664
```

2. The program has one or more bugs. How does the output suggest something is wrong?

We will now start debugging this program,

Before we begin running the program, we will need to set a breakpoint. A breakpoint is a programmer-specified point in the code where we would like the program to "pause" at while the program is running. Why would we want to do this? Using a debugger, we may examine

variables in this paused state which will give us important information about whether or not the program is working correctly.
You may set the breakpoint at the beginning of the program when you don't know where the program is wrong, or where to start. The program will run and stop where the first break point is. Then on, you use different controls such as step-in step-over etc. To set a breakpoint in VS, you **click the source code line** to highlight and **<F9>** or you may click a left side of source code window frame (it may depend on your IDE). The breakpoint toggles on/off.

Then select **DEBUG → Start debugging** to start your debugging journey. Once it stops at the breakpoint, you may start using <step into> <step out> <step over> etc. on the menu.
Notice that the contents of the string `input` has changed. Since strings are null-terminated, when `gets` reads a string, it copies the user's input into the array passed into the function and places a null terminator (denoted by \0) into the array when it reads a carriage return.

3. Using the techniques you have learned so far, explain what gets copied into `maxNumConsonant` after the line above is executed. Look at your windows^^!
   Explain what `input[0] - '0'` does.

Continue executing the code **line by line** (use <step over> usually, if you come across a strange code sections, use <step out> to get out of the pit.) until the following is reached.

```
generateSpellings (phone, maxNumConsonant);
```

(*You will be prompted to enter a phone number with dashes removed somewhere in the process. After the program has read the input phone number, you may want to verify that the appropriate information was read.*)

`generateSpellings` is a defined function within this program. We may be interested in watching what happens within the `generateSpellings` function. If we select <`step over`> at this point, we will go to the next instruction instead of going inside the `generateSpellings` function. In order to go inside `generateSpellings`, we can use the <step into command (or F11):

Now, look at the "Locals" window again^^. You may see another new world display all about generateSpelling stuff. A couple of lines at the top show the contents of the two variables passed into the function: `phone` and `maxNumConsonant`. Your output may be somewhat different.

For your information, `0x7fffbda8` is the hexadecimal (base 16) equivalent for 2,147,466,664. The memory addresses are denoted by the `0x` prefix. As you can see, the hexadecimal format is a more efficient way of representing large numbers.

4. What is the value stored in `phone`? (Hint: Everyone's answer will be different.) Why is `phone` represented by an address?

Continue using the <`step over`> or <`step into`> command until you see:

```
convert (phone, phone_int);
```

Let's <step into> this function as well:

```
phone=0x7fffbda8 "4460175"
phone_int=0x7fffbd08)

for (i = 0; i < 7; i++)
```

When **stepping into** a function, it does not display the contents of an integer array when it is passed in as an argument. This is unlike what it does with character arrays. (Note the difference between `phone` and `phone_int`.) However, regardless of variable type, it will always show you

what the value of that variable is when it is passed into the function. If this doesn't make sense, then you didn't answer question 4 correctly.

In the next five lines is a `for` loop. It is often helpful to monitor a `for` loop's progress. Watch the window 'Locals' (which means local variables).

Again, because values have not been assigned to the array cells above, their contents are in fact completely arbitrary. Your output may differ somewhat.

Monitoring the `for` loop's progress generally means stepping through it via repeated <step into> or <step over> command.

5. Use the techniques that you've learned thus far and determine whether or not the `for` loop in this function works. What evidence suggests so?

When you exit the function, you should see the following:

```
phone=0x7fffbda8 "4460175",
maxNumConsonant=4
```

The statement above lets you know that you've returned to the calling function (in this case, `generateSpellings`).

6. Starting on the above line, a series of nested loops is encountered. How does each nested loop work? Make sure you step through the loop and demonstrate that you've actually examined how the loop works.

Continue executing the code line by line until you reach

```
print_if_good(maxNumConsonant, word, &level);
```

7. What is the current value of `word`? Is its value correct?

8. The address of the variable `level` is passed into `print_if_good` as opposed to the variable's contents. The comments explain that this is a technique that allows the function to change the contents of a variable passed into a function. Explain how passing a pointer to a variable accomplishes this

Let's **<Step into>** this function: Now, you are inside

```
maxNumConsonant=4
word=0x7fffbd48 "ggm01pj",
level=0x7fffbd50 {0}
numVowel = -858993460
i = -85993460
```

In the following code, there is a `for` loop that counts the number of vowels and an `if` statement that determines whether or not the generated word is printed. We are interested in testing what happens when the `if` condition is true. Since we may not enter the condition every time, we could end up clicking `<step into/over>` many times before we reach the statements within the `if` condition. One way to skip to the "interesting" part of the program is to set the new break point where `printf ( "%s ", word);` within the `if` statement in `print_if good()`. Then select the command

<step out>

In this way, we have asked the program to stop at the new break point. You should eventually see something similar to

```
maxNumConsonant=4
```

```
word=0x7fffbd48   "iio01pj",
level=0x7fffbd5 {0}
numVowel=3
i=7
```

Print the word and go to the next line:

```
*level++;
```

Now find out what the value of `level` and `*level` is (your output will vary):

The hexadecimal number `0x7fffbd55` is 2,147,466,581 in decimal. Let's find out what happens when we execute this line.

```
if ((*level)%7 == 0)
```

The value of `level` is now `0x7fffbd54` or in decimal, 2,147,466,580. `*level` is still 0. What result did we expect? It seems like we might have a problem here.

9. Explain what the problem is in this section of code. Fix the problem in the code, and recompile the program. What happened to the warning?

When you've recompiled the program, verify the rest of the program works as described.

Checklist

A satisfactory grade on this assignment requires the following:

1. ***Questions*** *and answers* to boxed question 1 through 9 in a homework/project report format **hw01StudentID.doc or hwp file.** Refer to "How to write a report".
   Corrected version of **hw01StudentID.cpp.**
2. Post your source file and answer file into **hw1** folder in Piazza board. (Make sure that you post the files such that all instructors can see them all, but not to the public or students.
3. **Note**: a significant portion of the grade for this homework depends on what you write in response to those questions.

Submitting or posting your solution

- Make sure your code **compiles** and **runs** right before you submit it. Every semester, we get dozens of submissions that don't even compile. Don't make "a tiny last-minute change" and assume your code still compiles. You will not receive sympathy for code that "almost" works.
- If you only manage to work out the homework problem partially before the deadline, you still need to turn it in. However, don't turn it in if it does not compile and run.
- In this semester, I am trying to use the Piazza instead of the good ole^^ home-made Hisnet. The Piazza is well known bulletin board service which has been used recently in many universities including Harvard, Yale, Princeton, MIT, and so on. Maybe you are the first students who use the Piazza among universities in Korea or even in Asia. I have worked with Piazza team such that Piazza can accept two email domain names (handong.edu and hgu.edu). Post your files to me personally, not to open to everyone, in Piazza. Your feedback on your homework may be given to you through Piazza. Even if we have not used this kind of process before, let's give us a trial. If it does not work well, you may go back to good ole stuff^^.
- After submitting, if you realize one of your programs is flawed, you may edit or modify your posting as long as it is **before the deadline**. You will have to resubmit any related files together, even if you only change one. You may modify your posting as often as you like. **Only the last version** you submit before the deadline will be graded. **Never modify/edit your post after its deadline.** Then yours will be graded.(period)
- This will be the standard procedure for submitting future homework and projects as well.