University of Wollongong

School of Computer Science and Software Engineering (SCSSE)

CSCI124/MCS9124 Applied Programming Spring 2015

Lab 5 (1 mark)

Due at the end of your Week 6 Lab Class.

Background:

This will be a relatively short lab. The focus of the lab will be on using the `ddd` debugger. In addition to this you will get to investigate pointers and write a simple function using pointers.

Task 1:

Part 1

You have been given the source code for the program `selection.cpp`, which is supposed to implement selection sort. The selection sort program provided to you produces the incorrect result. Your job is to debug it.

1. Pick 10 random numbers (perhaps by running the program). Write these numbers down and show how selection sort, sorts them without using the computer i.e. pen and paper. You should show each iteration of the main loop – similar to how the program does it. In addition to this you should show for each iteration, the candidate elements that may be swapped. Place your findings in a file called lab5t1.txt.

2. Make sure the `selection.cpp` program is compiled with symbol table information. Run the program through the debugger (`ddd`) – compare how you think selection sort works with how the actual program works. Explain in detail (step by step) the process you engage in to understand how this program behaves. Place you findings and process in the file `lab5t1p1.txt`.

   HINT: When debugging you should try to understand the flow of control of the program and the values of its variables. Use the debugger to give you a better picture.

3. Once you have done this – edit the program and make suitable corrections so that it is able to sort a set of numbers. In doing so modify your program so that rather then generating 10 random numbers, it reads numbers up until EOF. You can assume your program is not to read in more than 100 numbers. Additionally give variables sensible names and document the code. Place your final solution in `selection.cpp`.

Part 2

In the same directory there is a program called `caesar.cpp`. The program when executed prompts for a key which is a character shift for the cipher. In addition to this the program takes the name of the file to

encrypt and the name of the file to store the encrypted output. If there is a problem with these files the program terminates.

The Caesar cipher works only on alphabetic characters i.e. `A – Z` and `a – z`.

If a file contains the string

```
CatzZ
```

And the key is 2 – then this should be encrypted as:

```
EcvbB
```

Run the program, and using `ddd`, work out why it does not behave in this way. Some of the problems ARE not apparent at all and require further investigation.… Document the steps you use (`lab5t1p2.txt`) and be sure to inspect the variables. Explain the problems using the information you gathered from `ddd`.

Once you have identified the problem – fix it. Place your solution in `caesar.cpp`.

Task 2:

In a file called `rotate.cpp` create three integer variables set to `'1' '2'` and `'3'`, and three character variables set to `'a', 'b'` and `'c'` as locals in main.
Print the address of each of these variables. You will need to use reinterpret_cast<void*>() to print the addresses of the characters, because cout interprets char* as the base address of a cstring.
Create an array of 3 integers initialized to 4, 5, 6. Print the array variable directly. Print the address of elements, 0, 1 and 2.

Examine the addresses (they are in hex by default). Add a block comment above the cout statements describing change between variables which were declared early and those declared later, and how integer and character variables change.

Write a function called `rotate` which returns void and takes three pointers to integer (by value). Like a swap function, `rotate` should rotate the values pointed to by those pointers. So if the addresses of variables with values `'1' '2'` and `'3'` are passed in, the variables should have values `'2' '3'` and `'1'` after rotate is called. Test that it is working by calling it on both the 3 integers you declared, and printing the values before and after the call. Do the same for the 3 array elements.

Submission

You are to submit the following files:

```
selection.cpp
caesar.cpp
rotate.cpp
lab5t1p2.txt
lab5t1p1.txt
```

You should submit all the above files using the submit program:

```
$ submit –c csci124 –a lab5 –u USERNAME FILES
```