REF:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*    Usage: Used  
\*    Title: Subtracting pointers  
\*    Author:   
\*    Date: 2014  
\*    Code version:   
\*    Availability: http://stackoverflow.com/questions/26366115/subtracting-pointers#comment41390775\_26366558

\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Use the Visual Studio Debugger to help answer the three questions below. In each case write an explanation of what is happening in each question. Upload your explanations using the link provided in Moodle.

Q1: The following code contains three serious memory bugs, when run it usually loops continuously until it either ends in a memory access violation fault or by using up all the available memory on the system. Identify, solve and explain each of the three bugs. If the programme still has run time errors you have not found all the bugs.

// this program is buggy

#include <iostream>

using namespace std;

void main()

{

double\* d = new double;

for(unsigned int i = 0; i < 3; i++) {

d[i] = 1.5 + i;

}

for(unsigned int i = 2; i >= 0; i--) {

cout << d[i] << endl;

}

}

* Main must return an int.
* You are not checking if the memory is full after you have created the pointer.
* You cannot have unassigned in the 2nd for loop when displaying the content of the memory address assigned to pointer d.
* You are not deleting the values in the memory adress that pointer d points to.
* You are not setting pointer d to NULL.

Q2: What does this program output and why?

void main()

{

char chr = 'N';

if (chr == 'Y' || 'y')

cout << "chr is y" << endl;

else if (chr == 'N' || 'n')

cout << "chr is n" << endl;

else

cout << "chr is something else" << endl;

}

* The Program outputs y.
* The program inputs ‘y’ because the else if statement is partially wrong. The else if statement is first checking if chr == ‘Y’. It then uses the ‘or’ statement (i.e. ||) but it is not checking any char input variable after the or statement. This problem is replicated towards the rest of the else if statement.
* You have not included the <iostream> and the using namespace std
* The main must return an int value.

Q3: The following program compiles and runs. It outputs 5 and 3. Explain in detailed steps what this program is doing.

#include <iostream>

using namespace std;

void main()

{

int t1[] = {0,0,1,1,1}, t2[] = {0,0,1,1,1};

int \*p1 = t1, \*p2 = t2;

while (!\*p1++ || !\*p2++);

cout << (p1-t1) << endl;

cout << (p2-t2) << endl;

}

Hint: Read Kernighan & Ritchie, The C Programming Language, 2nd Edition, Prentice Hall PTR, 1988, p. 166, Section A.7.7 Additive Operators, last paragraph.

Hint 2: Read Kernighan & Ritchie, The C Programming Language, 2nd Edition, Prentice Hall PTR, 1988, p. 48, Section 2.12 Precedence and Order of Evaluation (Table 2.1)

* The main program must return an int.
* The following outputs 5 and 3 because of the iterations in the while loop.
* The while loop is an empty statement and does nothing
* p1 is initially equal to &t1[0] . p2 is equal to &t2[0]. if one of the conditions in the while statement returns true, each Iteration causes an increase in &t1[0]/&t2[0] of either p1 or p2.
* The conditions in the while loop will only return true if !\*p1++ == 0 or !\*p2++ == 0 or if both conditions are true.
* If the 2 conditions in the while loop are returned false, the program exits the loop and incrementation is stopped.
* When consoled out, (p1 –t1 is equivalent to &t1[5] - &t1[0]) p1-t1 = 5.
* This is the same for p2 - t2 (p2 – t2 is equivalent to &t2[3] - &t2[0]) = 3.

Q4: Are the following two programs equivalent? Explain the behaviour and output from these two programs. Both programs compile and run.

Program (1)

#include <iostream>

using namespace std;

int main()

{

int count = 1;

for (; count <= 5 ; count++)

{

int count = 1;

cout << count << "\n";

}

return 0;

}

Program (2):

#include <iostream>

using namespace std;

int main()

{

int count = 1;

while (count <= 5)

{

int count = 1;

cout << count << "\n";

count++;

}

return 0;

}

* The two program may look the same but there are some difference:

1. When program one is executed,it first initialises the variable ‘count’ to 1 which is a variable accessible from anywhere inside the main().
2. For loop is then executed and the ‘count’ variable (accessible to anywhere within the main()) is passed in as an argument to check for a particular condition to enter the for loop.
3. If condition is true and it entered the for loop, it can creates a local variable called count. This variable is only accesible locally inside the for loop of program 1. It is initialised to 1 and is consoled out. The for loop increments the grobally accessed ‘count’ variable by 1 until the for loop is retured false and it exits the loop.
4. In Program 2, a variable called count(also accessible within the main()) is created and initialised to 1. This variable is then passed in as an argument inside the while loop. If the conditions inside the while loop is true, it enters it. Inside the while loop it is creating a new local variable called ‘count’ each time it enters while loop. This ‘count’ local varaible is initilised to 1 and consoled out. It is never incrementing the ‘count’ variable outside the while loop so ‘count’ will always be equals to 1. This will create an infinite while loop.