Computer Science 1MC3
Lab 4 – Borland C++ Compiler

In this lab we will learn how to code and run a simple program in C using the Borland C++ compiler. This is very important as you will not be able to do any programming without understanding how to execute your code. By the end of this lab activity you should be comfortable with the Borland compiler. If your not just ask your friendly T.A.

Step 1:

Open Texpad (*start→general campus applications→textpad*) and type the following code without the line numbers:

```
/1/          #include <stdio.h>
/2/
/3/          int main( int argc, const char *argv[] ) {
/4/
/5/                printf("Hello World.\n");
/6/                return 0;
/7/
/8/          }
```

Any program that you write will have this basic structure. Line 1 says you will be using the standard input/output library, line 2 indicates the start of the main program. The `printf` command on line 5 is the basic way to output something to the screen and line 6 just returns a null value since all functions must return something. You will write your code in between the open `{` and close `}`.

Save your notepad file to d:/temp, there should be a shortcut to this when you are prompted to save. When you save your file the reserved words in your program should turn a different color.

Step 2:

Open the Borland c++ compiler (*start→general campus applications→programming languages→Borland C++ 5.5*).

When opened type in (do not include the '>'):

```
> bcc32 thefilename.c your code will then compile, giving you any
error messages if there are errors.

> thefilename your program will run.
```

If there is no mistakes in your code you should see this on your screen:

```
Hello World
```

Otherwise you will receive an error message. The Borland C++ compiler will even supply the line number in which the mistake was made. Common mistakes include forgetting a semi-colon or misspelling a command. Either way the error messages are usually clear enough for you to find a fix your mistake.

If your code executed great! But in order to really understand the compiler you should go back and make a mistake on purpose to see what happens. So go back to step one and make an error.

For those people who made mistakes on their first attempt go back to step one and fix your error. If you can't find your error just call over your T.A. to help you.

Write a program that calculates your age by taking the year 2003 and subtracting from it the year in which you were born. Using printf print to the screen:

```
Hello, my name is Jane and I am 19 years old.
```

Design your program in such a way that I can walk up to your computer and change the year in your program and the whole thing will still work.

Write a program that takes a one-digit number and prints a series of lines, each repeating the digit one more time up to the number, and then each with one less digit back to one, making a triangular patter. For example, with and input of 5, the program would print:

```
5
5 5
5 5 5
5 5 5 5
5 5 5 5 5
5 5 5 5
5 5 5
5 5
5
```

Don't worry about taking input from a keyboard just make it so there is a variable x at the beginning of your code that you change before you compile.

Write a program, using a function that will except x and y and return the value of x^y. To start you off this is how the program looks like without any code in it.

```
int power(int x, int y);

int main (int argc, const char *argvp[]) {

        return 0;

}

int power (int x, int y) {

        return;

}
```
