Lab 11: Pointers

Pointers are variables that store the address of a memory location. They are called pointers because they ‘point to’ the memory location whose address they store. For example, if the pointer variable \( p \) stores the value B1A2 then it ‘points to’ memory location B1A2.

(Memory addresses are usually labeled using hexadecimal. Hexadecimal is simply a base 16 counting system. In a base 16 system we need symbols to stand for the values 10 through 15 and the letters A through F are used. So, B in hexadecimal = 11. For example, B1A2 = 11*16^3 + 1*16^2 + 10*16^1 + 2*16^0, which is \( 11*4096 + 1*256 + 10*16 + 2 = 45056 + 256 + 160 + 2 = 45474 \).)

Pointers are declared using a \(^*\). For example, the statement:

```c++
int* p; // C++ style
int *p;  // C style
```
creates a pointer (variable) called \( p \) that can point to a memory location storing an integer (it can also point to memory locations storing other types, but that will lead to big problems). In this case we haven’t made \( p \) point anywhere yet.

The following code illustrates using pointers:

```c++
int *p;   // create a pointer
int i = 5; // create an integer variable
p = &i;   // make p ‘point to’ i
*p = 6;   // change the value in the location p points to
cout << i; // prints _6_; i has ‘magically’ changed
cout << *p; // also prints 6
```

To make a pointer point to a regular variable we used the \& symbol (\( p = &i; \)). To access the value a pointer is pointing to we used the \(*\) symbol (\(*p = 6;\)). You can also change where a pointer points to, that is change what memory location it stores, by using operators like ++ (\( p++; \)). This would increase the memory location stored in the pointer, which makes the pointer point to a later memory location.

(The amount that ++ adds to a pointer’s location depends on the type the pointer is declared to point to. For example, if a pointer \( p \) points to chars then ++ typically changes the memory value stored by \( p \) by 1 byte because chars take up 1 byte in memory.)
Exercise 11.1

The following code demonstrates some of the different ways pointers can manipulate values in memory. When you compile and run the code you will see that the array is initially “abcdef” and ends up as “xxxxxx”. You need to figure out on which line each of the original letters is changed to an ‘x’.

```cpp
#include <iostream>
using namespace std;

void function1(char *q);

int main() {
    char array[] = "abcdef";
    char *p;
    cout << "initial array: " << array << endl;
    p = array;
    *p = 'x';
    p = &array[1];
    p++;
    *p = 'x';
    p = &array[3];
    p[2] = 'x';
    p--;
    func1(p);
    func1(array);
    func1(&array[3]);
    cout << "final array: " << array << endl;
}

void func1(char *q) {
    q++;
    *q = 'x';
}
```

You only need to turn in a list saying the line number that changed a given letter. For example, ‘the a was changed on line number 6.’ (which clearly isn’t correct). For letters changed in the function give the line number in main() where the function was called.