Pointer Arithmetic vs. Array Notation

This exercise is different from our usual exercises. You don’t have so much a problem to solve by creating a program but rather some things to understand about the programming language you are learning. To do so, we are going to write little programs which illustrate the relation between what we write and how it executes.

Iterating over an array of integers

Let’s consider the following program fragment;

```
int i , myarray[ SIZE ] ;
for ( i=0 ; i < SIZE ; i++ )
    myarray[i] = 0 ;
```

Use it to reset to 0, and then display, an integer array. Then, implement the same functionality by using pointer arithmetic notation instead of the array square brackets indexing scheme.

```
int i , myarray[ SIZE ] ;
for ( i=0 ; i < SIZE ; i++ )
     xxxxxxxxx = 0 ;
```

We are now using the pointer arithmetic notation but we are still accessing each array’s element by using *myarray*. Let’s use a pointer *p* instead; (1) initialize it to *myarray*, (2) increment it at each iteration, (3) use it to access each element in turn.

```
int i , myarray[ SIZE ] ;
int *p;
for ( i=0 , (1) ; i < SIZE ; i++ , (2) )
    (3)  = 0 ;
```

Find a way to rewrite this code and get rid of the index variable *i* you will stop iterating when *p* is pointing beyond the address of the last element of this array.

Iterating over a string

Let’s do something similar but with strings now.

```
int i;
char str [ 3 ] = {’a’, ’b’, ’\0’};
for ( i=0 ; str[i] ; i++ )
    putchar ( str[i] );
```

Rewrite the above program fragment so that is uses a pointer on *char* named *p* to iterate over the string. You will not use an index variable this time nor the SIZE of the string to decide when you’ll stop.