CSE 410 Section 1 – C and Pointers

C Introduction
C is syntactically very similar to Java, but there are a few key differences to watch out for!

```c
#include <stdio.h>
int main()
{
    // printf() displays the string inside quotation
    printf("Hello, World!");
    return 0;
}
```

This prints out Hello World to the standard output!

Pointers
C uses pointers explicitly. If we have a variable `x`, `&x` gives the address of `x` rather than the value of `x`. If we have a pointer `p`, `*p` tells us to use the value that `p` points to, rather than the value of `p`.

Consider the following declarations and assignments:

```c
int x;
int *ptr;
ptr = &x;
```

The unary operatory `&` returns the address of a particular variable. We can represent the above three lines of code graphically with:

![Memory Diagram](image)

x currently doesn’t contain a value since we did not assign x a value!

```
x = 5;
```

After executing the line above, the memory diagram becomes the following:

![Memory Diagram](image)

The variable `x` stores the value 5 and the variable `ptr` stores the address of `x`. Essentially, `ptr` “points” to `x`.

```
*ptr = 200;
```

After executing the line above, the memory diagram becomes the following:

![Memory Diagram](image)

We modified the value of variable `x` by dereferencing `ptr`!
Box and Arrow Diagram
Draw out the memory diagram by stepping through the function below:

```c
int main(int argc, char **argv) {
    int x = 351;
    int *p;       // p is a pointer to an integer
    p = &x;       // p now contains the address of x

    *p = 410;
    x = 333;

    return 0;
}
```

Step 1
- x
  - 351

Step 2
- p
- x
  - 351

Step 3
- p
- x
  - 351

Step 4
- p
- x
  - 410

Step 5
- p
- x
  - 333
Commenting C Code
The following functions have no comments. Document the code to prevent it from causing further confusion.

1. /* Returns the sum of the first N elements in ARR. */
   int foo(int *arr, size_t n) {
       return n ? arr[0] + foo(arr + 1, n - 1) : 0;
   }

2. /* Returns -1 times the number of zeroes in the first N elements of ARR. */
   int bar(int *arr, size_t n) {
       int sum = 0, i;
       for (i = n; i > 0; i--) {
           sum += !arr[i - 1];
       }
       return ~sum + 1;
   }

3. /* Does nothing. */
   void baz(int x, int y) {
       x = x ^ y;
       y = x ^ y;
       x = x ^ y;
   }

Programming with Pointers
Implement the following functions so that they perform as described in the comment.

1. /* Swaps the value of two ints initialized outside of this function. */
   void swap(int *x, int *y) {
       int temp = *x;
       *x = *y;
       *y = temp;
   }

2. /* Increments the value of an int initialized outside of this function by one. */
   void plus_plus(int *x) {
       (*x)++;
   // or: x[0]++;
   }
Fixing Logical and Syntactical Errors

The following code segments may contain logic and syntax errors. Find and correct them.

1. /* Returns the sum of all the elements in SUMMANDS. */
   int sum(int* summands) {
     int sum = 0;
     for (int i = 0; i < sizeof(summands); i++)
       sum += *(summands + i);
     return sum;
   }

2. /* Increments all the letters in the string STRING, held in an array of length N. Does not modify any other memory which has been previously allocated. */
   void increment(char* string, int n) {
     for (int i = 0; string[i] != 0; i++)
       *(string + i)++;
   }

3. /* Copies the string SRC to DST. */
   void copy(char* src, char* dst) {
     while (*dst++ = *src++);
   }

   // This code has no errors.