CSCI 101: Fundamentals of Computer Programming
Lab 5: Pointers
Fall 2005

Introduction

As of now, the functions learnt have a severe restriction in the sense that it can only return one value. Thus, if there is a need to have a function that returns multiple values, one would have to make multiple functions. This can be annoying and cumbersome. Thus, the introduction of this week's material – pointers.

Pointers are a new datatype that is somewhat an extension of the current known datatypes. Using pointers, functions can now indirectly change the values of multiple variables (as long as they have access to the pointers pointing to them). A pointer in essence usually contains the address of the variable it is pointing to. Initialization of pointers is best shown in the example below.

```
int var1 = 10;
int *pointer1;
pointer1 = &var1;
```

The first line of code above simply makes a new variable called `var1` and initializes it to `10`. The second line makes a new pointer called `pointer1`. The compiler recognizes that it is a pointer from the `*` sign in front of the pointer name. Hence, this is the reason why variable names CANNOT start with a `*`. This pointer's value is then set to the address of `var1` using the `&` sign on the third line. Access to the `var1` variable can now be done indirectly using the pointer. For example, `*pointer1 = 5;` would change the value of `var1` to `5`.

A good way for you to better understand pointers, is to make a simple program that simply initializes the values of the variables and pointers, and print them out. Try to answer the questions below to see if you fully understand pointers:

a) Is the value of `pointer1` the same as `*pointer1`?
b) If the value of `pointer1` is changed to `100`, what is the value of `var1`?
c) Why would the value of `pointer1` need to be set to `&var1`?
d) What would happen if `*pointer1` is set to `&var1`?

Pointers are generally simple concepts and extremely useful when used correctly. However, because it is a little indirect and has severe repercussions if used wrongly, it has generated a sense of fear around it. But as long as the foundations are set right, it should be a piece of cake.

Example

An example of a program that uses pointers is below. It changes the values of two variables without directly accessing them in the function `changeVal`. It firsts initializes two variables `x` and `y` to `0`, and makes two pointers that points to them respectively. The function `changeVal` then changes the values that the pointers are pointing to, thus indirectly changing the values of `x` and `y`. You can download this at *http://scf.usc.edu/~csci101/labs/lab6pointer.c*

```
#include <stdio.h>
void changeVal(int*, int*, int, int);

int main(){

  int x = 0, y = 0;
  int *xp, *yp;
  xp = &x;
  yp = &y;

  printf("Value of x before changeVal = %d\n", x);
  printf("Value of y before changeVal = %d\n", y);
  changeVal(xp, yp, 10, 100);
  printf("Value of x after changeVal = %d\n", x);
  printf("Value of y after changeVal = %d\n", y);

  return 0;
}

void changeVal(int *xp, int *yp, int valx, int valy)
{
  *xp = valx;
  *yp = valy;
}
```

Exercise

For your first pointer exercise, write a program to do the following. Parts (a) and (b) are to be done in two separate functions where you need to pass by reference. Parts (c), (d) and (e) are to be done in only **ONE** function where you will call the three subfunctions written from your previous lab. Reuse part (f) from your previous lab in this exercise.

a) Prompt the user for his/her name
b) Prompt the user to enter two coordinates in the Cartesian coordinate system. `(x1, y1)` & `(x2, y2)`
c) Calculate and outputs the distance of the two points given the two points.
d) Calculates and outputs the angle it makes with the horizontal.
e) Calculates and outputs the slope of the line.
f) Print out your outputs in the main function.

Show your completed program to your TA/LA to receive lab credit.