(Pointers and Functions)

1. Write a short C program that declares and initializes (to any value you like) a double, an int, and a char. Next declare and initialize a pointer to each of the three variables. Your program should then print the address of, and value stored in, and the memory size (in bytes) of each of the six variables.

Use the “0x%x” formatting specifier to print addresses in hexadecimal. You should see addresses that look something like this: "0xbfe55918". The initial characters "0x" tell you that hexadecimal notation is being used; the remainder of the digits give the address itself. Use the sizeof operator to determine the memory size allocated for each variable.g

1. Write a function that returns a pointer to the element with the largest value stored in an array.
2. Write a function to compute the distance between two points and use it to develop another function that will compute the area of the triangle whose vertices are **A(x1, y1)**, **B(x2, y2)**, and **C(x3, y3)**. Use these functions to develop a function which returns a value 1 if the point **(x, y)** lines inside the triangle ABC, otherwise a value 0.
3. Write a short C program that declares and initializes (to any value you like) a double, an int, and a string. Your program should then print the address of, and value stored in, each of the variables. Use the format string "%u" to print the addresses as unsigned (32-bit non-negative) integers.
4. Write a function operations that takes 5 parameters: first two are integer numbers and represent input data, and the last 3 parameters represent output data, being the sum, difference and product calculated from the first two parameters.
5. Write a function void shuffle\_ints(int\* array, int len); that shuffles the contents of the given array of integers. The parameter len contains the number of int values in the array. Write the shuffling code so that, given a random function with uniform distribution, every possible ordering of the contents is as likely. (Use the rand function provided by the standard C library.)

Example call:

int nums[] = { 0, 1, 2, 3, 4, 5, 6 };

shuffle\_ints(nums, 7);

Example result:

1, 0, 6, 4, 3, 2, 5

1. Find and eliminate all code that generates Segmentation Fault

#include<stdio.h>

int main()

{

char \*\*s;

char foo[] = "Hello World";

\*s = foo;

printf("s is %s\n",s);

s[0] = foo;

printf("s[0] is %s\n",s[0]);

return(0);

}

1. Find out (add code to print out) the address of the variable x in foo1, and the variable y in foo2. What do you notice? Can you explain this?

#include<stdio.h>

void foo1(int xval)

{

int x;

x = xval;

/\* print the address and value of x here \*/

}

void foo2(int dummy)

{

int y;

/\* print the address and value of y here \*/

}

int main()

{

foo1(7);

foo2(11);

return 0;

}

1. The program below uses pointer arithmetic to determine the size of a 'char' variable. By using pointer arithmetic we can find out the value of 'cp' and the value of 'cp+1'. Since cp is a pointer, this addition involves pointer arithmetic: adding one to a pointer makes the pointer point to the next element of the same type.

For a pointer to a char, adding 1 really just means adding 1 to the address, but this is only because each char is 1 byte.

1. Compile and run the program and see what it does.
2. Write some code that does pointer arithmetic with a pointer to an int and determine how big an int is.
3. Same idea – figure out how big a double is, by using pointer arithmetic and printing out the value of the pointer before and after adding 1.

#include<stdio.h>

int main( )

{

char c = ‘Z’;

char \*cp = &c;

printf("cp is 0x%08x\n", cp);

printf("The character at cp is %c\n", \*cp);

/\* Pointer arithmetic - see what cp+1 is \*/

cp = cp+1;

printf("cp is 0x%08x\n", cp);

/\* Do not print \*cp, because it points to memory not allocated to your program \*/

return 0;

}