**Labsheet 8**

**Pointers**

1. Write a program which contains one regular integer variable **x** and one pointer to integer **ip.** Try out the following basic uses of pointers:

* Assign **ip** the address of **x**.
* Using the pointer as the argument to **scanf()**, read a value into the location **ip** points at ( **i.e. x**) and then print out the value of **x.**
* Assign a new value to **\*ip**, and print out **x**.
* Assign a value to **x** and print out **\*ip**.

1. Write and test the following function:

**void CharSwap( char \*cp1, char \*cp2 );**

The function must swap the characters in the memory locations stored by its two pointer arguments.

1. It is inconvenient to read a lot of user input from inside **main().** It would be nice to move the job into a function. Functions can only return a single value so an input function would need to take several pointers as argument, one for each value to be read in from the user. These pointers could then be used in calls to **scanf()** to fill in the correct locations with the input.

Write an input function which reads in the user’s first name, age and income. The function should take 3 pointers as arguments, one pointer to an array of characters for the name, one pointer to **int** for the age, and one pointer to **float** for the income. This will allow the main program to determine if an input error occurred.

Write a main program to test the input function.

1. Write the following function:

**void input( int \*small, int \*medium, int \*large);**

The function must read in 3 integers from the user, assign the smallest of the three to **\*small**, the middle one to \***medium**, and the largest to \***large.** Write a main program which tests the function.

1. Write a function which finds both the minimum and maximum elements in an array of **doubles.** The function will need four arguments: the array, the length of the array, and two pointers to **double** through which the function can return the minimum and maximum values. Here is the prototype for the function:

**void MinMax( double \*data, int length, double \*retmin, double \*retmax);**

Write a main program to test the function.

1. Write your own version of the library function **strcat().** Recall that **strcat()** takes two arguments, **dest and src**, and appends **src** onto the end of **dest**. **strcat()** must first find the end of the **dest**  string, and then copy **src** onto the end of **dest.**

Declare the function like this:

**void StringCat( char \*dest, char \*src);**

1. Write a program which uses the **sizeof** operator to determine the number of bytes used by the various pointer types ( **char\*, int\*, float\*, double\*, void\* ).**

Even though pointers to different types are usually the same size, it is important to always use the correct type of pointer ( for example, always use an **int\*** to point at an **int** variable).

When a value is assigned to a variable through a pointer, the compiler uses the type of the pointer to determine how to store the value in memory. Different types are stored in very different formats. For example, floating point types are stored in “scientific notation” while integers are stored as an exact value. Using an incorrect pointer type will result in the data being stored in an incorrect format.

1. A pointer must not be dereferenced until it contains a valid address (i.e. until it is pointing at something). Write a program to see what happens on your system when an invalid pointer is dereferenced. To do this, declare an int\* and assign a value to what it points at (which is nothing in this case). Compile and run the program.