Jeremy Rios-Martinez

COMSC 265

Chapter 21, 22 and 23 assignment.

1. **Algorithms and Design Patterns.**

* What is a design pattern?

A design pattern is a general reusable solution to a commonly occurring problem in [software design](http://en.wikipedia.org/wiki/Software_design). A design pattern is not a finished design that can be transformed directly into [code](http://en.wikipedia.org/wiki/Code_(computer_programming)). It is a description or template for how to solve a problem that can be used in many different situations.

* How are algorithms and design patterns different and how are they similar?
* How are design patterns, code reuse?
* Why should you know about design patterns?

1. **Function Pointers.**

* Create an array of function pointers.

Store the pointers of the three following functions in the array.

Call the function using the array element, pass 3.14195 into the array.

What is the output?

void print\_stuff(float data\_stuff)

{

Cout << "Print stuff answer is " << data\_stuff << endl;

}

void print\_message(float data\_message)

{

cout << "Print Message answer is “ < data\_message << endl;

}

void print\_float float data\_boat)

{

cout << Print float answer is “ << data\_float<< endl;;

}

1. **TypeDef**

A first step towards making our own types is to set our own name to types.

The keyword "typedef" does this for us. Examples;

typedef short int16; /\* 16-bit integer \*/

typedef unsigned short uint16; /\* 16-bit unsigned integer \*/

typedef int int32; /\* 32-bit integer \*/

typedef unsigned int uint32; /\* 32-bit unsigned integer \*/

The above means that an alias "int16" is defined for the type "short," and an alias "uint16" is defined for the type "unsigned short," and so on for the "int" types.

* Is it good or bad programming to use TypeDef ?

1. **Enumeration** –

The easiest type to define is an enumeration.

Enumeration is a way of setting names to integer values.

A classic example of an enumeration type is the days of the week:

// Defintion of new type

enum DayNames { Sunday,

Monday,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday };

// Declaration using new type

enum DayNames dayOfWeek;

// If (DayOfWeek == Sunday )

The above example first defines an enumeration type called "enum DayNames", which includes the names of the days of the week.

The values for the names begin with 0 for "Sunday", 1 for "Monday" and so on through 6 for "Saturday."

* Write a program that proves the values of Sunday thru Monday are 0 to 6 in dayOfWeek.
* Write a program that trys to store an integer outside the bounds of 0 to 6 in dayOfWeek

Explain your results.

* Write a program that assigns the in sequence the day of the week to the dayOfWeek.

Use a loop to check which day of the week it is.

Print out the Meaning of the day of the week.