**Chris Dang Lab 3**

Look at the program geology.cpp to see what it does. Note the long sequence of const declarations that associate meaningful names with the numeric codes of the rocks.

1. Study the declarations and loops in the main() function. Now “reverse engineer” the program by writing an algorithm for main() in pseudocode. (See p. 16 of the text for an example of an algorithm written in pseudocode.)

**Part 1:**

**declare*sample***

**loop the following:**

**Display prompt to get rock from user**

**Store user input into *sample***

**If *sample* is equal to ending code (10) is entered, break.**

**Display what the rock is to the user**

**Part 2:**

**Display List of Rocks as header**

**Loop following:**

**Rockval is initialized to BASALT code. While Rockval is less than escape code (10)**

**Display rock code**

**Increment Rockval**

1. Now compile and execute the geology.cpp program.

What happens if you enter the name of a rock such as BASALT?

**An Infinite Loop occurs, spitting out garbage.**

Now try entering the number codes of basalt, dolomite, and granite. Describe the output produced.

After each iteration, it displays the number code without the rock name; after entering the exit code “10”, it shows the list of rocks with the ten number codes.

1. Now we want to modify Part 1 of geology.cpp as follows. Eliminate the const definition and replace them with the declaration of an enumerated data type called Rock. You might want to comment them out and use them as a guide to write the enumerated data type declaration.

When you’ve finished that, change the type of sample from int to Rock. Then recompile and execute the program, noting any errors that occur. What happened?

**There is a syntax error because the extraction operator does not know how to handle the datatype Rock.**

1. Replace the statement cin>> sample; with

Inttemp ;cin>> temp; //input an integer

Sample = Rock(temp); //cast integer to Rock type

Now recompile and execute program as you did before, entering the numeric codes for basalt, dolomite and granite. Describe any differences (if any between this program’s output and that in our earlier attempt in step 2.

**No differences were found.**

1. Now we are going to modify geology.cpp further. Specifically, change the type rockVal in the second for loop from int to Rock. Then attempt to recompile and execute the program. What happened?

**There was a compile error because the increment operator does not know how to handle datatype Rock of rockVal.**

1. In your current working directory, create a header file Rock.h containing:

The required opening documentation and The Rock enumeration declaration. You may copy and paste this from your earlier geology.cpp file. Then remove the enumeration declaration from your geology.cpp fileand add the line #include "Rock .h " to geology.cpp below the usingnamespacestd ; line Now recompile geology.cpp and make sure that you don’t have any new compilation errors, except for the ++ error we encountered in step 5. Check here when finished\_\_X\_\_

1. Modify the for statement in Part 2 of geology.cpp to use the next() function instead of the + + operator to move rockVal from one Rock value to the next:

For (Rock rockVal = BASALT; rockVal< ROCK\_OVERFLOW ;

rockVal = next(rockVal))

cout<<rockVal<< “ “ ;

Now we need to develop the next( ) function. As with all function development, we begin by formulating a specification. A good way to begin is to identify what things are to be passed to the function (if any) and what the function is to return or what other actions it is to carry out. For next() , we have:

Receives: A Rock value rockVal

Returns: The successor of rockVal

(The successor of the last enumerator SALE is ROCK\_OVERFLOW.)

1. Create and add a prototype and your particular documentation for the next() function to the Rock.h header file.

Check here when finished\_\_X\_\_

1. Complete definition of next() and put in Rock.cpp. Why don’t the statements associated with the switch()statement’s cases in the next() function require a break; statement concluding the case implementations? **Because it is a function, the return portion will end the function while also returning a value.**

10. Now, compile, link and execute your geology.cpp program. Describe behavior below.

**Program runs the same as before**

11. You are now going to enhance the Rock data type by adding a new operation—determining whether a

rock is igneous, metamorphic, or sedimentary. Specifically, you are to add a new function kind () to

type Rock that returns one of the strings “igneous,” “metamorphic,” or “sedimentary,” indicating the

kind of rock, given that:

Basalt, granite, and obsidian are igneous.

Marble, and quartzite are metamorphic.

Dolomite, limestone, gypsum, sandstone, and shale are sedimentary.

You can use the next ( ) function as a model. Before writing the kind() function you should

consider what information it needs and what information is should return.

What information must you send to your kind () function?

**The value of Rock**

What information will your k in d () function return?

**It will return a string; either igneous, metamorphic, or sedimentary**

Check here when finished adding kind() and adding <string> to rock.h\_\_\_x\_\_\_

Check here when finished adding a loop for kind()\_\_\_x\_\_\_