Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Given the declarations:**

enum MovieRatings {G, PG, PG13, R, X};

MovieRatings thisMovie;

Using the following code, write the **for loop expression** to correctly loop through all values of the MovieRatings enum and print properly. Assume the switch and output compile without errors.

for (**thisMovie = G; thisMovie <= X; thisMovie = MovieRatings(thisMovie + 1**) {

switch(thisMovie) {

case G:

cout << “The movie is rated G” << endl;

break;

case PG:

cout << “The movie is rated PG” << endl;

break;

case PG13:

cout << “The movie is rated PG13” << endl;

break;

case R:

cout << “The movie is rated R” << endl;

break;

case X:

cout << “The movie is rated X” << endl;

break;

}

}

1. **Structures.**
2. Create a structure called **PersonRec** with the following members. Use the proper variable DATATYPES for the members:

First name

Last name

age

weight

height (inches)

**struct PersonRec{**

**string firstName;**

**string lastName;**

**int age;**

**int heightInches;**

**float weight;**

**};**

1. Create an array variable of type **PersonRec** that can hold 50 components.

**PersonRec fiftyPeople[50];**

1. Write the code segment to read from a file into the structure array. Declare an input file stream identifier for this purpose. Open the file and **read until end of file**. Keep a count of how many rows actually read. The data file contains the following data:

Hermione Granger 22 105 65

Tony Stark 44 185 74

Alan Turing 109 162 70

**Ifstream read;**

**Int countRow = 0;**

**Int I = 0;**

**Read.open(whateverthenameofthefilehostingtheaboveinformationis);**

**While(Read >> fiftyPeople[i].firstName >> fiftyPeople[i].lastName >> fiftyPeople[i].age >> fiftyPeople[i].heightInches >> fiftyPeople[i].weight){**

**countRow++;**

**i++;**

**}**

1. Using a looping construct, output to the console (cout) the data read into the structure array. Make sure you do not go outside the bounds of the array or beyond the number of rows **actually read that was maintained during the file read**.

**For(int I = 0; I < countRow; I++){**

**Cout << fiftyPeople[i].firstName << “ “ << fiftyPeople[i].lastName << “ “ << fiftyPeople[i].age << “ “ << fiftyPeople[i].heightInches << “ ” << fiftyPeople[i].weight << endl << endl;**

**}**

1. You do not need to write any #include statements, using statements and you do not have to write code to open the file. Assume the file is open.
2. You **do** need to declare variables needed for the program segment.
   1. **Write the specification (i.e., what would be in a header file) for a class called bankAccount. Code the data member section (as private) and the function prototypes (as public). You do not need to write any include, using or other directives. You do not need to write function code. Include the following:**

Data:

balance

Functions:

Default constructor

deposit

withdraw

get balance

**class bankAccount{**

**public:**

**account(float);**

**float deposit();**

**float withdraw(); // the idea would be that these functions return the amount inputted while inside the function and then return that value to the main to alter the balance;**

**private:**

**float balance;**

**};**

* 1. **Create an object of bankAccount using the default constructor.**

**bankAccount bankOfAmericaHoldings(30092.12);**

1. **Consider the following linked list:**

first

4

9

32

78

5

struct NodeType {

int val;

NodeType\* next;

}

NodeType\* first;

Write a traversal loop that **sums** the values of the linked list structure member **val** starting at the beginning of the list to the end and outputs (cout) the sum (you only have to write the cout statement, not the actual sum). **Define any additional variables needed**.

Ensure the beginning of the list is preserved in NodeType\* first. You do not need to write any #include, using or main statements. You do not need to create or insert any other nodes into the list.

**Int sum = 0;**

**Int \*ptr;**

**Ptr = first;**

**While(ptr != NULL){**

**Sum += Ptr ->val;**

**Ptr = ptr -> next;**

**}**

**Cout << sum;**

1. **Pointers and dynamic memory.**
   1. **Given the declarations what does statement 1 do?**

float x;

float\* ptr;

ptr = &x;

\*ptr = 24.9; // statement 1 **The value stored at the address pointed to by pointer is updated from what it was before to 24.9.**

* 1. **Write a statement that dynamically allocates memory for a float and returns the address into the pointer variable declared in part a.**

**Float someFloat;**

**ptr = &someFloat;**

* 1. **Assign the value 12 to the memory dynamically allocated in part b and pointed to by the pointer variable in part a.**

**\*ptr = 12;**

**// I think this is what you’re asking for**