Date

CISC 205 – OOPS C++

Professor Larry Forman

Telephone: 619.388.3666

E-Mail: [LForman@sdccd.edu](mailto:lforman@sdccd.net)

Mail Box: Room A-8

Office: BT-210-G

**TRAINING ASSIGNMENT #2.2C: GETTING CLASSY**

**=== DUE ===**

**=============**

**TASKS:**

**0 –** First, read this Task Sheet!

Second, put a check mark by each Task number **and** letter when you complete it.

**1 –** **READ**: **HO#1.1Color, HO#1.3, HO#2.1, TDB**

**2 –** **TA OBJECTIVES**

* Define a class with member variables and functions
* Incorporate a menu-based C++ program to deploy the class functions
* Maintain effective Michelangelo documentation

**3 – SPECIFICATIONS**

Design a class, LarrysMover, to model a character – namely, ASCII code \2 (or your choice) – moving up, down, left and right across the screen within a designated “play area”. BTW: Feel free to name the class and any member items

any names you feel are (more) reasonable than the ones I have indicated.

1. Include in the class the following member variables:
2. The current column position of the character
3. The current row position of the character
4. A (boolean) “flag” to indicate whether or not the character moves with or without a “trail”

**STAR: Also include a member variable to represent ANY choice of ASCII character**

**STAR: Also include a color for the character**

**STAR: Expand the previous STAR to include a member variable indicating the color for the character**

**STAR: Represent the “character” NOT by one symbol but by, say, a 5 column by 3 row box in which some pattern/shape appears, like:**

**\_[ ]\_**

**(“)**

**(\_:\_)**

**STAR: Expand the previous STAR by making the two-dimensional character/shape as a member variable (using an array or a vector)**

**STAR: Expand the previous STAR by using a pointer – not an array or vector.**

**STAR: Create and use 4 “reasonable” characters which appear depending upon which direction the character is moving, e.g. left, right, up or down**

**STAR: Expand previous STAR to include a 5th “reasonable” character which appears when the character is not in motion**

**STARS (2+): Expand either of the two previous STARS where, say, a 5 by 3 box will represent the different directional characters**

**STAR: Also include a member variable for the “health” status of the character**

**STAR: Also include other “game-oriented” status counts as member variables for the character like power, etc.**

**STAR: Also include a “flag” to indicate whether or not the character moves with a “tail” of 3 dots: . . . horizontally and vertically.**

**STAR: Expand previous STAR to work when turning “corners“ so “tail” turns as character moves up and down**

**STAR: Add a member variable to indicate the length of the “tail” for either of the two previous STARS**

**STAR: Add a series of member variables in which some reasonably should be public and some private. In your program comments provide a valid justification for your public and private choices**

**STAR: Also include a string member variable that contains the file name of a short (annoying) sound that will play every time the character moves**

1. Include the following as the class functions and justify in your program comments why you made any of them public and any private. *Also, for each function be sure to incorporate “error-checking” to prevent functions from crashing the program by setting reasonable default values or taking appropriate actions*:
2. gotoXY( int x, int y) that moves the cursor to column x and row y.
3. gotoXY( int x, int y, string message) that moves cursor to position x, y where the message is displayed
4. moveUp that moves the character up one row but stops when reaches the “ceiling”. *Note: For all the “moves”, replace the character in the “old” position with a “ ” (space) UNLESS the “trail” flag is “on” in which case do not “erase” the old character*
5. moveDown that moves the character down one row but stops when reaches the “floor”
6. moveLeft that moves the character one column to the left but stops when reaches the left “wall”
7. moveRight that moves the character one column to the right but stops when reaches right “wall”

**STAR: Play a short sound every time the character moves**

**STAR: Expand previous STAR to include a total of 4 short sound files, where a different sound will play based on which of the four directions the character is moving**

**STAR: Play a different short sound when the character hits a “border” – the “ceiling”, “floor” and left and right “walls”.**

**STAR: Expand previous STAR to play a different sound for each of the four borders**

**STAR: When any of the four moves tries to go beyond the “boundary”, display just beyond the boundary a warning/sarcastic message like “No Way!” or “Huh?” or “Ouch!” or whatevah you fancy**

**STAR: Create a “flag” or some overloaded move functions that enable player to “wrap-around” when moving beyond a boundary. Like, if reach far right “wall” and deploy moveRight, then the character stays in the same row but moves to the first column inside the far left wall. Presumable, while playing a “game” the player can select a designated key on the keyboard to invoke this “wrap-around” feature.**

**STAR: Also include four functions that take care of moving diagonally one row and one column in all four diagonal directions, like moveUpLeft will move character one row up and one row to the left, etc.**

**STAR: Add overloaded “move” functions that have an integer parameter to indicate the number of rows or columns the character will move**

**STAR: Replace the four move functions – moveUp, etc. – with one function that uses a parameter to indicate the direction**

**STAR: Expand the previous STAR to additionally work for the four diagonal directions**

1. getRow that – duh – gets the current row of the character
2. getColumn that gets the current column of the character
3. setRow that – duh – sets the current row of the character
4. setColumn that sets the current column of the character
5. jump function with two parameters, one for the column and one for the row of where the character directly jumps to without trails/tails
6. displayPlayArea that displays the enclosed area where your character “moves”. This area should be 21 columns wide and 13 high with reasonable symbols for the perimeters of the “box” (see the Extended ASCII Table handout to use symbols like \310 or alternatively \xC8 AND \311 or alternatively \xC9, etc. for the corners and sides of a “box”)

**STAR: Add an overloaded displayPlayArea function with parameters for the coordinates of the upper left corner of the play area along with the width and height of the play area**

**STARS (2+): In the displayPlayArea function add “obstacles” that act like “walls” to prevent character from moving past them**

**STAR: Add a displayCurrentPlayStatus function that displays the current state of all the member variables in an effective way**

1. Default constructor that initializes the member variables to your preferred default values and performs any actions you feel make sense

**STAR: Add an overloaded constructor to do something “interesting”**

1. Include a simple menu-based system in a program in which you declare an object of your class and deploy all your class methods in an “interesting” way to test that your entire class “works”

**STAR: Have one of your menu selections deploy a function that uses your class functions to have your character start from the upper-left corner of the play area and walk around the perimeter of the play area two times. The first time there will be a “trail” and the second time there will not be (so the trail will get “erased” during the second tour)**

**STAR: Have one of your menu selections deploy your class functions to display a “countdown” of 3, 2, 1 and 0 via an “LED” type pattern like:**

**XXXX**

**X**

**XXXX**

**X**

**XXXX**

**XXXX**

**X**

**XXXX**

**X**

**XXXX**

**X**

**X**

**X**

**X**

**X**

**XXXX**

**X X**

**X X**

**X X**

**XXXX**

**STAR: Have another menu selection to have the character create a trail of “nested” zeros, one inside of another until it stops**

**STAR: Have another menu selection to create a “spiral” pattern that starts in the center of the play area with the character spiraling around and out to the perimeter**

1. Have a farewell/exit menu selection that displays a friendly farewell message as well as:

The current date and time

Your complete ID INFORMATION (nicely formatted),

CREDITS and any STARS you did in the usual enumerated format (with credit given only for STARS that are completed and fully displayed here)

STARS

1. Blah-blah-blah
2. More blah

TOTAL STARS = 2

<<< Hit ENTER to continue >>>

**4** – **SAVE** your file early and often -- like every 5 minutes. And, use your backup "disk"

**5** – **TEST** your file early and often -- like every 5 minutes -- How do you eat an apple?

**6** – **PROGRAM DOCUMENTATION**

1. First, include via comments your complete ID INFORMATION, PROGRAM DESCRIPTION, **CUSTOM-DEFINED FUNCTION LIST** and CREDITS (to those who helped you and whom you helped).
2. **Include "inline credits" to acknowledge specifically where you were helped.**
3. Add comments immediately before each segment of your program to describe "highlights" of coming attractions. Insert at least one blank line before each of these comments. Make all function, variable and const names self-descriptive, clear and fully formed (no abbreviations or secret code names). **Use verbs for function names, like “displayXyz” or “getXyz” or “calculateXyz” or “doXyz” and nouns for constants and variables.**
4. Define a const for your name and use it whenever your name appears
5. Add “banners” for: your prototypes to include description of each function, all constants, variables, start of function definitions, end of function definitions and in the function definitions with a description of each function (you can use the same ones as in your prototypes). See Handout #1.2 for all the details.
6. **FUNCTION PROTOTYPES – In your function descriptions, be sure to include how the parameters are used and for non-void functions identify what information will be returned. Make sure to say “return x” for non-void functions. For example, consider this hypothetical function prototype:**

**//NAME: getFavoriteNumber**

**//DESCRIPTION: Prompt user by name for favorite number & return it**

**int getFavoriteNumber( string userName )**

1. IMPORTANT: Embrace the Michelangelo structure for **every** C++ program.

**7** – **DEMO (= beta testing)** your program in the Lab with a completed TASK SHEET

**8** – **HAND IN HARDCOPY** of your accurate TIME SHEET

**9** – **STARS**

1. In a NEW engaging way, use >=4 windows.h colors throughout
2. Implement this assignment by creating and using a custom header file to hold all your function prototypes and definitions. Include your Michelangelo documentation with ID INFO, etc. in each header file
3. Add extra pizzazz and briefly explain what you did:
4. Effectively use three new "Advanced" features (= not yet introduced). List them:
5. Demo before the due-date (N.B.: You still can do more STARS on due-date)

**“**[**Creativity requires the courage to let go of certainties**](http://www.brainyquote.com/quotes/quotes/e/erichfromm151839.html)**.”**

**Erich Fromm**

**“Creativity is intelligence having fun.”**

**Albert Einstein**