SHORT REPORT

All the .c files for the task are given.

Code description

main.c -

- “LinkedList* storemove; “- declares a global variable storemove of type LinkedList* to store history of the movement.

- The command-line arguments argc and argv are passed to the main function. It verifies that the appropriate number of arguments is given before opening the supplied file. If the file cannot be opened, the software terminates and an error message is shown.

- By calling the createLinkedList function to make an empty linked list, it initializes the movement history.

- The code enters a game loop that continues until the box reaches the goal position. If the move is 'u' and undo is enabled, the undo feature will be shown where it will go back to its previous position.

- The code then handles the different move cases: up, down, left, and right. It checks if the player can move to the new position without hitting obstacles or pushing boxes. If it's a valid move, it updates the map and player/box positions accordingly.

- The game loop then continues until the box reaches the goal location after updating and printing the map. The code activates the system's buffer at the end of the game and prints a thank-you message, then the allocated memory will be free.

map.c -

intializeMap: this method allows to dynamically create a game map. Along with the starting locations of the player and the objective, it also considers the number of rows and columns on the map.

printmp: prints the game's color map along with the map itself. It requires as inputs the map, the color_map, the number of rows, and the number of columns. The setCellBackground and setBackground functions are used to set the background color of cells, and the map is printed within (*).

UpdateTheMap: Based on a player's motion, this function updates the game map and color map. It accepts as inputs the map, the color_map, the number of rows and columns, the player's old position, and their new position. It moves the player to the new location, updates the color_map. The color trail implementation was also done here but it did not work properly.

validity: This function checks if a move is valid within the game map. The move position is first checked to see if it is outside the map's limits, and if it is, 0 is returned. The move position is next examined to see if it contains an obstacle ('O').

free_map: This function frees the memory allocated for the game map.

loadMap: This function loads a game map from a file. The filename, pointers to variables for the number of rows and columns, and pointers to variables for the starting location of the player are all inputs. It reads a file that has been opened and tests to see if it failed to open. It retrieves the file's row and column counts.

create_colorMap: This function creates a color map for the game map.

free_colorMap: This function frees the memory allocated for the color map.

random .c - the functions are the same as it was given with the assignment.

terminal.c - contains disable buffer and enable buffer (this was given with the assignment)

color.c - sets the background colors of the map.

How to compile,

gcc main.c map.c random.c terminal.c color.c linkedlist.c

MAKEFILE,

For compilation and linking command for the final executable file “box”.

Header Files,

map.h - header file for main.c and map.c,

random.h - header file for random.c,

terminal.c - header file for terminal.c,

color.h - header file for color.c

DOES THE CODES RUN?

Overall, the whole game works but not entirely on question 3.5 (on assignment question paper) where I have to implement a way to display a color trail but it is not visible in my implementation of codes. So, the code implementation of 3.1, 3.2, 3.3, 3.4, 3.6 works except for 3.5 where the color trial needs to be displayed.