**Source Code Print-out**

/\* FILE: main.c

\* Manhattan Tourist Problem (the game)

\* Author: Jason Ang Chia Wuen

\* Last Modified: 13/5/2018 1:10pm

\*/

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

/\* Function to read textfile "records.txt" \*/

void readTextFile()

{

FILE \*textfile;

int name[50];

int score;

textfile = fopen("records.txt", "r");

if (textfile == NULL) /\* Checking if text file can be opened \*/

{

printf(" \*No record of previous players found\*");

}

{

while( fscanf(textfile, "%s %i", &name, &score) == 2) /\*'==2' because name and score are two items \*/

{

printf("\n %s %i\n", name, score);

}

}

fclose(textfile);

}

/\* Function to add name and score into textfile "records.txt" \*/

void writeTextFile(char name[], int score)

{

FILE \*textfile;

textfile = fopen("records.txt", "a+"); /\*append (doesn't delete previous data)\*/

if (textfile == NULL) /\* Checking if text file can be opened \*/

{

printf("Error opening file.\n");

exit(1);

}

fprintf(textfile, "\n%s", name);

fprintf(textfile, "\n%i", score);

fclose(textfile);

}

/\* Function to initialize array with random numbers \*/

void initialize\_array(int array[][8], int \*score)

{

int x;

int y;

int rand\_x;

srand((unsigned)time(NULL));

for (y = 0; y < 8; y++)

{

for (x = 0; x < 8; x++)

{

array[x][y] = rand()%10; /\* Number in array randomly generalized \*/

array[0][0] = 0; /\* First number will always be zero \*/

if (((x == 0) && (y == 0)) || ((x == 7) && (y == 7)))

{

printf("-|%i| ", array[x][y]);

}

else

{

printf("- %i ", array[x][y]);

}

}

if(y == 7)

{

printf("\n\n"); /\* Last row has no pattern \*/

}

else

{

printf("\n | | | | | | | | \n"); /\* Prints pattern below every number row \*/

}

}

(\*score) = (\*score) + array[0][0];

}

/\* Function to display player's movement in the array \*/

void move\_array(int array[][8], int real\_x, int real\_y, int trace\_x[], int trace\_y[])

{

int x;

int y;

system("cls"); /\* Clears screen \*/

for (y = 0; y < 8; y++)

{

for (x = 0; x < 8; x++)

{

if (((x == 0) && (y == 0)) || ((x == 7) && (y == 7)) || ((x == real\_x) && (y == real\_y)) || ((x == trace\_x[0]) && (y == trace\_y[0])) || ((x == trace\_x[1]) && (y == trace\_y[1])) || ((x == trace\_x[2]) && (y == trace\_y[2])) || ((x == trace\_x[3]) && (y == trace\_y[3])) || ((x == trace\_x[4]) && (y == trace\_y[4])) || ((x == trace\_x[5]) && (y == trace\_y[5])) || ((x == trace\_x[6]) && (y == trace\_y[6])) || ((x == trace\_x[7]) && (y == trace\_y[7])) || ((x == trace\_x[8]) && (y == trace\_y[8])) || ((x == trace\_x[9]) && (y == trace\_y[9])) || ((x == trace\_x[10]) && (y == trace\_y[10])) || ((x == trace\_x[11]) && (y == trace\_y[11])) || ((x == trace\_x[12]) && (y == trace\_y[12])) || ((x == trace\_x[13]) && (y == trace\_y[13])))

{

printf("-|%i| ", array[x][y]);

}

else

{

printf("- %i ", array[x][y]);

}

}

if(y == 7)

{

printf("\n\n");

}

else

{

printf("\n | | | | | | | | \n");

}

}

}

/\* Function of player playing the game (background process) \*/

int process(int real\_x, int real\_y, int \*score, int input\_valid, char input, int array[][8])

{

int trace\_x[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /\* Previous position of array (x) \*/

int trace\_y[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0}; /\* Previous position of array (y) \*/

int moves = 0;

while ((real\_x != 7) || (real\_y != 7))

{

printf("\nNumber of attractions visited so far : %i\n", (\*score));

printf("\nEach number in the map represents the number of attractions along each street");

printf("\nYou may only move to the right or move down!\n");

printf("\nPress 'd' to go right");

printf("\nPress 's' to go down");

printf("\nPress 'q' to quit the game");

input\_valid = 0;

while (input\_valid == 0)

{

printf("\nPlease choose your next action:");

input = getch(); /\* Removes the need for player to press "Enter" key \*/

if (input == 'd')

{

if (real\_x == 7)

{

printf("\nInput error (You cannot go right anymore).\n"); /\* Prevents user from going beyond border \*/

}

else

{

input\_valid = 1; /\* Correct input \*/

real\_x++;

trace\_x[moves] = real\_x;

trace\_y[moves] = real\_y;

(\*score) = (\*score) + array[real\_x][real\_y];

move\_array(array, real\_x, real\_y, trace\_x, trace\_y); /\* Displays array movement for player \*/

moves++;

}

}

else if (input == 's')

{

if (real\_y == 7)

{

printf("\nInput error (You cannot go down anymore).\n"); /\* Prevents user from going beyond border \*/

}

else

{

input\_valid = 1; /\* Correct input \*/

real\_y++;

trace\_x[moves] = real\_x;

trace\_y[moves] = real\_y;

(\*score) = (\*score) + array[real\_x][real\_y];

move\_array(array, real\_x, real\_y, trace\_x, trace\_y); /\* Displays array movement for player \*/

moves++;

}

}

else if (input == 'q')

{

input\_valid = 1; /\* Correct input \*/

}

else

{

printf("\nIncorrect input, please try again"); /\* Incorrect input \*/

}

}

if (input == 'q')

{

break; /\* Player quits game \*/

}

}

return (\*score);

}

/\* Function to allow replay option \*/

void replay\_option(int \*status, int \*score, char name[])

{

int input\_valid = 1; /\* Validation \*/

char input;

while (input\_valid == 1)

{

printf("\nDo you wish to save your name and score? Press 'y' for yes and 'n' for no:");

input = getch();

if (input == 'y')

{

printf("\nYour name and score will now be recorded.\n");

writeTextFile(name, \*score); /\* Adds name and score of player into text file \*/

input\_valid = 0; /\* Validation ok \*/

}

else if (input == 'n')

{

printf("\nYour name and score will not be recorded.\n");

input\_valid = 0; /\* Validation ok \*/

}

else

{

printf("\nPlease input correctly.\n");

input\_valid = 1; /\*Validation not ok \*/

}

}

input\_valid = 1; /\* Reset validation for 2nd option \*/

while (input\_valid == 1)

{

printf("\nDo you wish to replay this game? Press 'y' for yes and 'n' for no:");

input = getch();

if (input == 'y')

{

printf("\nThe game will now repeat.\n");

(\*score) = 0; /\* Reset score \*/

input\_valid = 0; /\* Validation ok \*/

}

else if (input == 'n')

{

input\_valid = 0; /\* Validation ok \*/

(\*status) = 0; /\* Change variable to end program \*/

printf("\nThanks for playing!!! See you again...\n");

}

else

{

printf("\nPlease input correctly.\n");

input\_valid = 1; /\* Validation not ok \*/

}

}

}

typedef struct{

int x;

int y;

int score;

char input;

int input\_valid;

}gameData;

/\* Main function \*/

int main()

{

gameData data = {0, 0, 0};

char name[20];

int input\_stat=1 /\* Variable to play game again \*/, array[8][8];

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\* WELCOME TO Manhattan Tourist Problem! \*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Previous Players' Record \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n Previous Player(s) Score(Attractions Visited)\n");

readTextFile(); /\* Prints previous text file data \*/

printf("\nPlease key in your given name:");

scanf("%s", &name);

printf("\*\*\*Good day %s, let's start the game...all the best!!!\*\*\*\n", name);

while (input\_stat == 1)

{

printf("Press any key to continue...\n\n");

getch(); /\*detects any keyboard input (Windows only)\*/

initialize\_array(array, &data.score); /\* Initializes array with random numbers \*/

data.score = process(data.x, data.y, &data.score, data.input\_valid, data.input, array);

printf("\nCongratulations! You have completed your tour!\n");

printf("\nTotal attractions you have visited = %i\n", data.score);

replay\_option(&input\_stat, &data.score, name); /\* Give player option to save data and replay game \*/

}

return 0;

}