// SNAKE GAME

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* ALI GAUHAR,NAEEM UR REHMAN,RAAHIMA \*

// \* DSA-LAB PROJECT \*

// \* ooooooooooooooooooooooo 0 Oooooo \*

// \* o o \*

// \* o o \*

// \* o ooooooooooooooo \*

// \* o o \*

// \* oooo \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// header files which are included are//

//....................................//

#include <iostream> // input output function for console

#include <conio.h> // input output function from device like getch

#include <cstdlib> // library for memory allocation of linked list

#include <time.h> // to store processor time helps in providing seed for random

#include <windows.h> //for including the functions like sleep for slowing down the speed of game

using namespace std;

////////////////////////////////////////////////////////////////////////////////////////////////

//global food structure//

//.....................//

struct food

{

int x, y; // axis position of the food in the matrix

char ch; // character of food representation O

};

struct snake

{

int x, y; // axis position of the snake in the matrix

char ch; // character of head representation O

struct snake \*next; // for next node position to make a linked list

};

//defining the keys ascii//

//.......................//

#define up 72

#define down 80

#define left 75

#define right 77

//declaring some global variables//

//...............................//

int height = 20; // defining height of the map

int width = 20; // defining width of the map

int pos = left; // game will be started from the left position

bool out = false; // out variable

int score = 0; //calculating score

int level = 1; //specify the level

int speed; // to specify the high or low speed

int speedspecify; //for user input speed

int pcspeed = 1000; //to deal with pc speed increase it to slow down the snake speed

struct food f; //saves food position

// snake game consist of some function.

//....................................//

void incr(struct snake \*s); // incrementing snake length while food is eaten

void genfood(struct food \*f); // generating new food

void maper(char board[20][20]); // map drawing

void out\_snap\_code(); //shows the snape of how the player was out and his score

void win\_snap\_code(); //winning case snap

void swap(struct snake \*s, int c, int r, int t, int a, int b); //for swaping position after input

void self\_bit(struct snake \*s); //out on bitting itself

void poschange\_and\_boundary(struct snake \*s, int c); // changing snake position with respect to the input and swaping new positions

void level\_identifier(struct snake \*s); //identifies the level

void thrd\_level\_maper(char MAP[20][20]); //maping third level

void scnd\_level\_maper(char MAP[20][20]); //maping scond level

void frst\_level\_maper(char MAP[20][20]); //maping first level

void congrats(); //winning or losing quotes

void page1(); //display welcome screen

void page2(); //diaplay instrcution

void speed\_specification(); //next page asks speed and starts the game

struct snake \*Baby\_snake(char MAP[20][20], struct food \*f); //creates a baby snake thats position is on the mid of the screen

//defining functions from here//

//............................//

void win\_snap\_code()

{

Sleep(pcspeed);

out = true;

system("cls");

cout << "\n\n\n\n\n\n\n\n\n\n\t You Win" << endl;

Sleep(pcspeed + 2000);

system("cls");

cout << "\n\n\n\n\n\n\n\n\n\n\t YOUR SCORE: " << score << endl;

cout << "\n\n\n\n\n\n\n\n\n\n\t Showing snap" << endl;

Sleep(pcspeed + 5000);

system("cls");

}

void out\_snap\_code()

{

Sleep(pcspeed);

out = true;

system("cls");

cout << "\n\n\n\n\n\n\n\n\n\n\t OUT" << endl;

Sleep(pcspeed + 2000);

system("cls");

cout << "\n\n\n\n\n\n\n\n\n\n\t YOUR SCORE: " << score << endl;

cout << "\n\n\n\n\n\n\n\n\n\n\t Showing snap" << endl;

Sleep(pcspeed + 5000);

system("cls");

}

void congrats()

{

if (score == 140)

cout << "\n\n\n\n\n\n\n\n\n\n\t \*\*\*CONGRAGULATION\*\*\* \n\n\n\n\n\n\n\n\n\n"

<< endl;

else

cout << "\n\n\n\n\n\n\n\n\n\n\t \*\*\*GOOD LUCK FOR NEXT TIME \n\n\n\n\n\n\n\n\n\n"

<< endl;

}

void page1()

{

cout << "\t\t \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \n";

cout << "\t\t / \\ / \\ \n";

cout << "\t\t / /~~~~~\\ \\ / /~~~~~\\ \\ \n";

cout << "\t\t | | | | | | | | \n";

cout << "\t\t | | | | | | | | \n";

cout << "\t\t | | | | | | | | / \n";

cout << "\t\t | | | | | | | | // \n";

cout << "\t\t (o o) \\ \\\_\_\_\_\_/ / \\ \\\_\_\_\_\_/ / \n";

cout << "\t\t \\\_\_/ \\ / \\ / \n";

cout << "\t\t | ~~~~~~~~~ ~~~~~~~~ \n";

cout << "\t\t ^ \n";

cout << "\n\n";

cout << "\t Welcome To The Snake Game! \n";

cout << "\t------This Game has been created by...-----------\n";

cout << "\t @ Ali Gauhar (02-134202-006) \n";

cout << "\t @ Naeem Ur rehman (02-134202-053) \n";

cout << "\t @ Raahima Irfan (02-134202-009) \n";

cout << "\tTo implement the Data structures taught by out lab\n";

cout << "\tteacher Miss Saba imtiaz and theory teacher miss Lubna...\n";

cout << "\tIt is an open source project anyone can make contribution\n";

cout << "\t Github link will be provided in the report\n";

Sleep(3 \* pcspeed + 14000);

system("cls");

}

void page2()

{

cout << "----------------------------------------------------instruction-----------------------------------------------------" << endl;

cout << "----------------------Level 1 is without walls, you will need 100 score to pass the level---------------------------" << endl;

cout << "-----------------------Level 2 is with walls , you will need 120 scores to pass the level---------------------------" << endl;

cout << "---Level 3 has 2 walls between the free space and side are without walls, 140 scores are needed to pass the level---" << endl;

cout << "------------------------------------------Use arrow keys to play the game-------------------------------------------" << endl;

cout << "---------------------------------------------press e for ending the game--------------------------------------------" << endl;

cout << "----------------------------------\*\*\*speed of the game depends on the pc speed\*\*\*-----------------------------------" << endl;

Sleep(3 \* pcspeed + 12000);

system("cls");

}

void speed\_specification()

{

cout << "\n\n\n\n\n\n\tSpecify the speed\n\n\n";

cout << "\n\n\tHere the low has been set to 0 and high has been set to 1000\n\tself-specify Speed can be set between 0 to 1000\n";

cout << "\tspeed ?...press 1 for high, 2 for low, 3 for self specify: " << endl;

cout << "------>";

cin >> speed;

if (speed == 1)

{

speedspecify = 1000;

}

else if (speed == 2)

{

speedspecify = 0;

}

else if (speed == 3)

{

cout << "Higher the number more will be the speed" << endl;

cout << "Now specify the speed-->";

cin >> speedspecify;

cout << endl;

}

for (int i = 5; i > 0; i--)

{

system("cls");

cout << "\n\n\n\n\n\n\n\n\n-----------------------------------------------------------------------------------------------------------" << endl;

cout << "----------------------------------------Game will be started soon------------------------------------------" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*>>>\t" << i << "\t" << endl;

Sleep(pcspeed);

}

}

struct snake \*Baby\_snake(char MAP[20][20], struct food \*f)

{

for (int i = 0; i < height; ++i) //printing of snake map

{

for (int j = 0; j < width; ++j)

MAP[i][j] = ' '; //taking the empty space in all area

}

f->x = 4; // by default x position of food

f->y = 4; // by default y position of food

f->ch = 'Q'; // Q for recognizing food in the map

struct snake \*temp = (struct snake \*)malloc(sizeof(struct snake)); // Temporary Node for holding nodes and creating the nodes

struct snake \*temp1 = (struct snake \*)malloc(sizeof(struct snake)); // Temporary Node for holding nodes and creating the nodes

struct snake \*s = (struct snake \*)malloc(sizeof(struct snake)); // Node for Head of the snake

///////////////////////first picture of the snake will be specified here///////////////////////////////////////////////////

// Game will start from the center

//Head specifications

temp1->x = height / 2; //alloting center for the x in the map

temp1->y = width / 2; // alloting center of the y in the map

temp1->ch = 'O'; // O for recognizing the head of the snake in the map

temp1->next = NULL; //initially making it null further it will be joined with the segment

s = temp1; // savin the created head in the node s

temp->x = height / 2; //alloting center for the x in the map

temp->y = width / 2 + 1; //alloting center for the y+1 in the map....the next position of the Head

temp->ch = 'o'; // o for recognizing the segments of the snake

temp->next = NULL; //initially making it null further it will be joined with the next segment

temp1->next = temp; // saving the segments value in the next of the head where temp1->next is the next pointer of the head

temp1 = temp1->next; // accessing the last position which is the segment next to head for operating its pointer for the last node

temp = (struct snake \*)malloc(sizeof(struct snake)); // Allocating new memory to the temp which is used for segments next to the snake head

temp->x = height / 2; //alloting center for the x in the map

temp->y = width / 2 + 2; //alloting center for the y+2 in the map....the next position of the segment next to the head

temp->ch = 'o'; // o for recognizing the segments of the snake

temp->next = NULL; //initially making it null further it will be joined with the next segment

temp1->next = temp; // saving the segments value in the next of the head where temp1->next is the next pointer of the segment next to the head

temp1 = temp1->next; // accessing the last position which is the segment next to segment next to head for operating its pointer for the last node

temp = (struct snake \*)malloc(sizeof(struct snake)); // Allocating new memory to the temp which is used for segments next to the snake head

temp->x = height / 2; //alloting center for the x in the map

temp->y = width / 2 + 3; //alloting center for the y+3 in the map....the next position of the segment next to the first segment

temp->ch = 'o'; // o for recognizing the tail of the snake

temp->next = NULL; // making the last position null because we nedd no more segment to e alligned till the food is eaten

temp1->next = temp; // saving the segments value in the next of the segment where temp1->next is the next pointer of the segment next to the last segment

return s;

}

void frst\_level\_maper(char MAP[20][20])

{

for (int i = 0; i < width + 2; ++i) ////////printing the upper wall

cout << "-";

cout << endl;

for (int i = 0; i < height; ++i)

{

cout << "|"; ////////////////////////printing the left wall

for (int j = 0; j < width; ++j)

cout << MAP[i][j]; //printing the map with its element

cout << "|"; ////////////////////////printing the right wall

cout << endl; //changing the row

}

for (int i = 0; i < width + 2; ++i)

cout << "-"; ////////////////////////printing the bottom wall

cout << endl;

cout << "Scores : " << score << endl;

cout << "Level : " << level << endl;

Sleep(pcspeed - speedspecify); //slowing down the speed according to the user desire

}

void scnd\_level\_maper(char MAP[20][20])

{

for (int i = 0; i < width + 2; ++i) ////////printing the upper wall

cout << "\*";

cout << endl;

for (int i = 0; i < height; ++i)

{

cout << "\*"; ////////////////////////printing the left wall

for (int j = 0; j < width; ++j)

cout << MAP[i][j]; //printing the map with its element

cout << "\*"; ////////////////////////printing the right wall

cout << endl; //changing the row

}

for (int i = 0; i < width + 2; ++i)

cout << "\*"; ////////////////////////printing the bottom wall

cout << endl;

cout << "Scores : " << score << endl;

cout << "Level : " << level << endl;

Sleep(pcspeed - speedspecify); //slowing down the speed according to the user desire

}

void thrd\_level\_maper(char MAP[20][20])

{

for (int i = 0; i < height; ++i)

{

for (int j = 0; j < width; ++j)

if (j == 3 \* width / 4)

MAP[i][j] = '\*';

}

///////////////////////////this is for the top wall iside///////////////

for (int i = 0; i < height; ++i)

{

for (int j = 0; j < width; ++j)

if (i == 2 \* width / 6)

MAP[i][j] = '\*';

}

//////////////////////////////////////////////////////////////////////

for (int i = 0; i < width + 2; ++i) ////////printing the upper wall

cout << "-";

cout << endl;

for (int i = 0; i < height; ++i)

{

cout << "|"; ////////////////////////printing the left wall

for (int j = 0; j < width; ++j)

cout << MAP[i][j]; //printing the map with its element

cout << "|"; ////////////////////////printing the right wall

cout << endl; //changing the row

}

for (int i = 0; i < width + 2; ++i)

cout << "-"; ////////////////////////printing the bottom wall

cout << endl;

cout << "Scores : " << score << endl;

cout << "Level : " << level << endl;

Sleep(pcspeed - speedspecify); //slowing down the speed according to the user desire

}

void level\_identifier(struct snake \*s)

{

if (score == 100)

level = 2;

else if (score == 120)

level = 3;

else if (score == 140)

{

win\_snap\_code();

}

/////////////////////////////////out the player if snake bit itself//////////////////////////////////////////////////////////////

//this is snake head

struct snake \*tempp = s;

self\_bit(s);

}

void self\_bit(struct snake \*s)

{

struct snake \*tempp = s;

int outx = tempp->x; //holding head x

int outy = tempp->y; //holding head y

tempp = tempp->next; //working on segments

while (tempp != NULL)

{

if (tempp->x == outx && tempp->y == outy) //if the head of axis becomes equal to any axis of the segments then out

out\_snap\_code();

tempp = tempp->next;

}

}

void swap(struct snake \*s, int c, int r, int t, int a, int b)

{

struct snake \*temp = s;

r = temp->x; // holding the snake head x axis

t = temp->y; // holding the snake head y axis

int p; // for holding x axis of n-1 th position

int q; // for holding y axis of n-1 th position

while (temp->next != NULL) // loop from the begning(head) of the snake to the end of the snake(these are now the segments beacuse head is new position which is x and y)

{ //loop is n-1

//here we are swaping the nth segment with n-1 where n is the segment next to head

p = temp->next->x; // holding the value of x into p of the segment on which the value of segmwent next to it is to be saved(segment 1 value)

q = temp->next->y; // holding the value of y into q of the segment on which the value of segmwent next to it is to be saved

////////////saving the position into its previous position and position is holded into the r and t

temp->next->x = r; // saving the first segment x position into second segment x(head saved in segment 1)

temp->next->y = t; // saving the first segment y position into second segment y

temp = temp->next; // incrementing the linked list

r = p; //holding the segment position x into r on which the value of its next segment is over-written

t = q; //holding the segment position y into t on which the value of its next segment is over-written

} //till the n-1 because the last value is to be deleted

temp = s; //taking temp as head of the snake

//head of the snake are not used for saving any values only segments are used

temp->x = temp->x + a; //saving the new position into head x

temp->y = temp->y + b; //saving the new position into head x

}

void incr(struct snake \*s)

{

struct snake \*temp = (struct snake \*)malloc(sizeof(struct snake)); // Making temporary variable for saving snake head position

temp->x = s->x; // head x axis saved

temp->y = s->y; // head y axis saved

temp->ch = 'X'; // incremented segments will be shown as o

temp->next = s->next; //getting the next segment of the snake head in temporary node

s->next = temp; //now snake head and second last position is denoting the same position

poschange\_and\_boundary(s, pos); // sending data to be operated

// This function makes the copy of the snake head and saves it in the second last position e.g head and next to head is same....

// giving new position to the head will not effect the snake look and length is incremented also

// since there are two same values at the head and segment next to head therefore in swaping the first segment which is the segment next to head and the second segment will have the same

// and because of this the added segment due to fruit will not be included untill the swaping is done till the tail (as one of the two same position becomes the last position and lost due to n-1 swaps)

// at the time of segment increment one segment of the tail lost "for only one cycle" as the food segment takes its position and the last two nodes are same values

// we are talking about the script change(s,pos); ->line number 68

// incase the fruit is not eaten then no addition in length and no headache of same values saved at multiple nodes

}

void poschange\_and\_boundary(struct snake \*s, int c)

{

//Defining variables to be used

// here c is the new input and pos is the previous input

// c variable is taken from parameter

// pos is globally defined variable

int a = 0; // a for increment and decrement in height

int b = 0; // b for increment and decrement in width

int r; // for holding x axis of nth position

int t; // for holding y axis of nth position

/////////////////////////////////////////////Deciding position with respect to input given////////////////////////////////////////////////

if (pos == left && c == right) // if going to the the left and user desire for right then continue to left inorder to avoid hitting itself

c = left;

else if (pos == right && c == left) //if going to the right and user desire for left then continue to right inorder to avoid hitting itself

c = right;

else if (pos == down && c == up) //if going to the down and user desire for up then continue to down inorder to avoid hitting itself

c = down;

else if (pos == up && c == down) //if going to the up and user desire for down then continue to up inorder to avoid hitting itself

c = up;

else if (c != left && c != down && c != right && c != up) // if no input then continue without changing

c = pos;

//////////////////////////////////////////////making position strategy with respect to input given//////////////////////////////////////////

pos = c; //saving c in pos inorder to make new input saved as previous

if (pos == left) //left

b = -1; // this wil be added into the snake head y axis to move left e.g y axis will be decremented

else if (pos == down) //down

a = 1; // this wil be added into the snake head x axis to move right e.g x axis will be incremented

else if (pos == right) //right

b = 1; // this wil be added into the snake head y axis to move right e.g y axis will be incremented

else if (pos == up) //up

a = -1; // this wil be added into the snake head x axis to move right e.g x axis will be decremented

/////////////////////////////////////////////////swapping for snake segments and head////////////////////////////////////////////////////////

//this is snake head

struct snake \*temp = s;

swap(temp, c, r, t, a, b);

//////////////////////////////////////////////////////Dealing with the boundaries////////////////////////////////////////////////////////////

if (level == 1)

{ //////////////without walls

//for x Coordinates

if (temp->x == -1) //if hit the Top then again appear from bottom

temp->x = height - 1;

else if (temp->x == height) //if hit the bottom then again appear from top

temp->x = 0;

//for y coOrdinates

else if (temp->y == -1) //if hit the left wall then again appear from right where x=width-1

temp->y = width - 1;

else if (temp->y == width) //if hit the right wall then again appear from left where x=0

temp->y = 0;

}

else if (level == 2)

{ // with walls

if (temp->x == 20 || temp->x < 0 || temp->y == 20 || temp->y < 0)

{ //IF HIT THE WALL THEN TERMINATE THE GAME(walls=width/height+2)

//out the player if snake hit the wall

out\_snap\_code();

}

}

else if (level == 3)

{ //no border walls // only center walls

//for x Coordinates

if (temp->x == -1) //if hit the Top then again appear from bottom

temp->x = height - 1;

else if (temp->x == height) //if hit the bottom then again appear from top

temp->x = 0;

//for y coOrdinates

else if (temp->y == -1) //if hit the left wall then again appear from right where x=width-1

temp->y = width - 1;

else if (temp->y == width) //if hit the right wall then again appear from left where x=0

temp->y = 0;

//if hits the center walls then out

//right wall

if (temp->y == 3 \* width / 4)

{ //if head hit the right wall then out

out\_snap\_code();

}

//top wall

if (temp->x == 2 \* height / 6)

{ //if head hit the top wall

out\_snap\_code();

}

}

}

void genfood(struct food \*f)

{

srand(time(0)); // givin seed to the random function

f->x = rand() % width; // generating the x access of the food with in the height which is 20

f->y = rand() % height; // generating the y access of the food with in the height which is 80

score = score + 10;

if (level == 3)

{

while (f->y == 3 \* width / 4 || f->x == 2 \* height / 6)

{ // if the food is generated on the center walls in level 3 then again generate

srand(time(0)); // givin seed to the random function

f->x = rand() % width; // generating the x access of the food with in the height which is 20

f->y = rand() % height; // generating the y access of the food with in the height which is 80

}

}

}

void maper(char MAP[20][20], struct snake \*s, struct food \*f)

{

struct snake \*temp = s; // Making temperory variable to operate the snake positions taking it as the snakes head

MAP[f->x][f->y] = f->ch; // saving randomly generated food on the map

int c = pos; //for getting new value in c

while (temp != NULL) //loop from the head to the tail

{ //saving the positions of the segments in the map

MAP[temp->x][temp->y] = temp->ch;

temp = temp->next;

}

for (int i = 0; i < height; ++i)

{

for (int j = 0; j < width; ++j)

cout << MAP[i][j]; //Drawing the map with its elements

cout << "\n"; //changing the rows

}

/////////////////////////////////////////////////////clear the board///////////////////////////////////////////////////

temp = s; //again taking the temp as head

while (temp != NULL) //loop from the head to the tail

{

MAP[temp->x][temp->y] = ' '; // removing elements data from the map(clearing the map) so not to overwrite and to remove the last mark

temp = temp->next;

}

MAP[f->x][f->y] = ' '; //removing previous fruit to make room for new

///////////////////////////////////////////////////////////////////////////////////Main loopof iteration///////////////////////////////////////////////////////////////////

while (!out)

{ //////////////////////////////////////////////////////////////////////////////////loop terminate the game//////////////////////////////////////////////////////////////////

///////////////////////////////Taking input from the user to go further for operation///////////

temp = s; // saving head of the snake in the temp

if (kbhit()) // checking if there are any inputs

c = getch(); // then catch the input in c

if (c == 'e') // press e to terminate

break;

poschange\_and\_boundary(temp, c); // for changing the position of the snake in the map according to the input given and dealing with the boundaries

MAP[f->x][f->y] = f->ch; //saving fruit on the map

///////////////////identifying the level with respect to scores

level\_identifier(s);

///////////////////////////////clear the screeen after out//////////////////////////////////////

system("cls");

if (f->x == s->x && f->y == s->y) // if the axis of the head of the sanke becomes equal to the axis of the goal

{

genfood(f); // then generate new food

incr(s); // and increment the snake length

}

////////////////////////saving segments on the map//////////////////////////////////////////////

while (temp != NULL)

{

if (temp->x >= 0 && temp->y >= 0)

{

MAP[temp->x][temp->y] = temp->ch; // saving segments of snake on the map

}

temp = temp->next;

}

///////////////////////level 1 printing/////////////////////

if (level == 1)

{

frst\_level\_maper(MAP);

}

///////////////////////////////level 2 printing///////////////////////

else if (level == 2)

{

scnd\_level\_maper(MAP);

}

///////////////////////////////level 3 printing//////////////////////////

else if (level == 3)

{

//////////////////////////saving mid walls//////////////////////////

//////////////////////////this is for right wall inside///////////////

thrd\_level\_maper(MAP);

}

///////////////////////////////////////////////////////clear the board////////////////////////////////////////////////////////////

temp = s;

while (temp != NULL)

{

if (temp->x >= 0 && temp->y >= 0)

{

MAP[temp->x][temp->y] = ' '; // removing elements data from the map(clearing the map) so not to overwrite and to remove the last mark

}

temp = temp->next;

}

MAP[f->x][f->y] = ' '; //removing previous fruit to make room for new

}

}

///////////////////////////Main Function Started/////////////////////////////////////

int main()

{

//////////////////////////////////////////////////Welcoming Screen//////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////contributions/////////////////////////////////////////////////////

page1();

///////////////////////////////////////////////////////instructions////////////////////////////////////////////////////

page2();

/////////////////////////////////////////////////////Speed Specifications//////////////////////////////////////////////

speed\_specification();

/////////////////////////////////////////////////////////////starting of game///////////////////////////////////////////////////////////////////////////

char MAP[20][20]; // map for the area in wich the snake will play

struct food f; //first food

struct snake \*s = Baby\_snake(MAP, &f); // fisrt defined snake with head at center position

///////////////////////////////////////////////////////go for the function to get into loop///////////////////////////////////////////////////////

//main function that handles the further game/functions

maper(MAP, s, &f);

Sleep(pcspeed + 4000); //pause after out or win

//clear after out or win

system("cls");

/////////////////////////////congrats on winning//////////////////////////////////////

congrats();

//////////////////////////////////////Game End///////////////////////////////////////

return 0;

}