1. Mini Project in C Snake Game

Code:-

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <conio.h>  // Define the dimensions of the game board  #define WIDTH 20  #define HEIGHT 20  // Define the directions in which the snake can move  #define UP 0  #define RIGHT 1  #define DOWN 2  #define LEFT 3  // Define the initial position of the snake  int snake\_x[100] = {WIDTH / 2, WIDTH / 2, WIDTH / 2};  int snake\_y[100] = {HEIGHT / 2, HEIGHT / 2 - 1, HEIGHT / 2 - 2};  int snake\_length = 3;  // Define the position of the food  int food\_x, food\_y;  // Define the direction in which the snake is moving  int direction = RIGHT;  // Define the game state  int game\_over = 0;  // Define the function to initialize the game  void init\_game()  {  // Initialize the snake position  snake\_x[0] = WIDTH / 2;  snake\_y[0] = HEIGHT / 2;  snake\_x[1] = WIDTH / 2;  snake\_y[1] = HEIGHT / 2 - 1;  snake\_x[2] = WIDTH / 2;  snake\_y[2] = HEIGHT / 2 - 2;  snake\_length = 3;  // Initialize the food position  food\_x = rand() % WIDTH;  food\_y = rand() % HEIGHT;  // Initialize the direction  direction = RIGHT;  // Initialize the game state  game\_over = 0;  }  // Define the function to draw the game board  void draw\_game\_board()  {  // Clear the screen  system("cls");  // Draw the top border  for (int i = 0; i < WIDTH; i++)  {  printf("#");  }  printf("\n");  // Draw the left and right borders  for (int i = 0; i < HEIGHT; i++)  {  printf("#");  for (int j = 0; j < WIDTH - 2; j++)  {  printf(" ");  }  printf("#\n");  }  // Draw the bottom border  for (int i = 0; i < WIDTH; i++)  {  printf("#");  }  printf("\n");  }  // Define the function to move the snake  void move\_snake()  {  // Get the new head position of the snake  int new\_x = snake\_x[0];  int new\_y = snake\_y[0];  // Move the head of the snake in the current direction  switch (direction)  {  case UP:  new\_y--;  break;  case RIGHT:  new\_x++;  break;  case DOWN:  new\_y++;  break;  case LEFT:  new\_x--;  break;  }  // Check if the snake has eaten the food  if (new\_x == food\_x && new\_y == food\_y)  {  // Increase the length of the snake  snake\_length++;  // Generate a new food position  food\_x = rand() % WIDTH;  food\_y = rand() % HEIGHT;  }  else  {  // Move the tail of the snake  for (int i = snake\_length; i > 0; i--)  {  snake\_x[i] = snake\_x[i - 1];  snake\_y[i] = snake\_y[i - 1];  }  }  // Update the head of the snake  snake\_x[0] = new\_x;  snake\_y[0] = new\_y;  }  // Define the function to check if the snake has collided with itself  int is\_game\_over()  {  // Check if the snake has collided with itself  for (int i = 1; i < snake\_length; i++)  {  if (snake\_x[0] == snake\_x[i] && snake\_y[0] == snake\_y[i])  {  return 1;  }  }  // Check if the snake has collided with the walls  if (snake\_x[0] < 0 || snake\_x[0] >= WIDTH || snake\_y[0] < 0 || snake\_y[0] >= HEIGHT)  {  return 1;  }  // The game is not over  return 0;  }  // Define the function to handle input from the user  void handle\_input()  {  // Get the input from the user  char input = getch();  // Update the direction of the snake  switch (input)  {  case 'w':  direction = UP;  break;  case 'd':  direction = RIGHT;  break;  case 's':  direction = DOWN;  break;  case 'a':  direction = LEFT;  break;  }  }  // Define the function to run the game  void run\_game()  {  // Initialize the game  init\_game();  // Draw the game board  draw\_game\_board();  // While the game is not over  while (!game\_over)  {  // Handle input from the user  handle\_input();  // Move the snake  move\_snake();  // Draw the game board  draw\_game\_board();  // Check if the snake has collided with itself  game\_over = is\_game\_over();  }  // Game over  printf("Game over! Your score is %d\n", snake\_length - 3);  }  // Define the main function  int main()  {  // Run the game  run\_game();  return 0;  } |

1. Mini Project Bank Management System.

Code:-

|  |
| --- |
| #include <stdio.h>  #include <string.h>  struct Account {  int accNumber;  char accHolderName[50];  float balance;  };  void checkBalance(struct Account acc) {  printf("Account Number: %d\n", acc.accNumber);  printf("Account Holder Name: %s\n", acc.accHolderName);  printf("Balance: %.2f\n", acc.balance);  }  void deposit(struct Account \*acc, float amount) {  acc->balance += amount;  printf("Amount %.2f deposited successfully!\n", amount);  checkBalance(\*acc);  }  void withdraw(struct Account \*acc, float amount) {  if (amount <= acc->balance) {  acc->balance -= amount;  printf("Amount %.2f withdrawn successfully!\n", amount);  } else {  printf("Insufficient balance!\n");  }  checkBalance(\*acc);  }  int main() {  struct Account myAccount = {123456, "John Doe", 1000.00};  int choice;  float amount;  do {  printf("\nBank Management System\n");  printf("1. Check Balance\n");  printf("2. Deposit\n");  printf("3. Withdraw\n");  printf("4. Exit\n");  printf("Enter your choice: ");  scanf("%d", &choice);  switch (choice) {  case 1:  checkBalance(myAccount);  break;  case 2:  printf("Enter amount to deposit: ");  scanf("%f", &amount);  deposit(&myAccount, amount);  break;  case 3:  printf("Enter amount to withdraw: ");  scanf("%f", &amount);  withdraw(&myAccount, amount);  break;  case 4:  printf("Thank you for using Bank Management System!\n");  break;  default:  printf("Invalid choice. Please try again.\n");  }  } while (choice != 4);  return 0;  } |

3.

|  |
| --- |
| #include <stdio.h>  #include <conio.h>  #include <windows.h>  #define WIDTH 20  #define HEIGHT 20  #define UP 72  #define DOWN 80  #define LEFT 75  #define RIGHT 77  int gameOver;  int score;  int tailX[100], tailY[100];  int tailLength;  int headX, headY, fruitX, fruitY;  int direction;  void Setup() {  // Initialize game variables  gameOver = 0;  score = 0;  headX = WIDTH / 2;  headY = HEIGHT / 2;  fruitX = rand() % WIDTH;  fruitY = rand() % HEIGHT;  tailLength = 0;  }  void Draw() {  // Clear screen  system("cls");  // Draw top boundary  for (int i = 0; i < WIDTH + 2; i++) {  printf("#");  }  printf("\n");  // Draw game area  for (int i = 0; i < HEIGHT; i++) {  for (int j = 0; j < WIDTH; j++) {  if (j == 0)  printf("#");  if (i == headY && j == headX)  printf("O");  else if (i == fruitY && j == fruitX)  printf("F");  else {  int isTail = 0;  for (int k = 0; k < tailLength; k++) {  if (tailX[k] == j && tailY[k] == i) {  printf("o");  isTail = 1;  }  }  if (!isTail)  printf(" ");  }  if (j == WIDTH - 1)  printf("#");  }  printf("\n");  }  // Draw bottom boundary  for (int i = 0; i < WIDTH + 2; i++)  printf("#");  printf("\n");  // Print score  printf("Score: %d\n", score);  }  void Input() {  if (\_kbhit()) {  switch (\_getch()) {  case UP:  direction = 1; // UP  break;  case DOWN:  direction = 2; // DOWN  break;  case LEFT:  direction = 3; // LEFT  break;  case RIGHT:  direction = 4; // RIGHT  break;  case 'x':  gameOver = 1;  break;  }  }  }  void Logic() {  // Update tail positions  int px = tailX[0];  int py = tailY[0];  int px2, py2;  tailX[0] = headX;  tailY[0] = headY;  for (int i = 1; i < tailLength; i++) {  px2 = tailX[i];  py2 = tailY[i];  tailX[i] = px;  tailY[i] = py;  px = px2;  py = py2;  }  // Move head according to direction  switch (direction) {  case 1: headY--;  break;  case 2: headY++;  break;  case 3: headX--;  break;  case 4: headX++;  break;  default:  break;  }  // Wrap around boundaries  if (headX >= WIDTH) headX = 0;  else if (headX < 0) headX = WIDTH - 1;  if (headY >= HEIGHT) headY = 0;  else if (headY < 0) headY = HEIGHT - 1;  // Check for collision with tail  for (int i = 0; i < tailLength; i++) {  if (tailX[i] == headX && tailY[i] == headY)  gameOver = 1;  }  // Check for collision with fruit  if (headX == fruitX && headY == fruitY) {  score += 10;  fruitX = rand() % WIDTH;  fruitY = rand() % HEIGHT;  tailLength++;  }  }  int main() {  // Initialize game  Setup();  // Game loop  while (!gameOver) {  Draw();  Input();  Logic();  Sleep(100); // Sleep for 100 milliseconds  }  return 0;  } |

4.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #define SIZE 4  #define WIN\_NUMBER 2048  int board[SIZE][SIZE];  int score;  void initializeGame() {  // Initialize board  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  board[i][j] = 0;  }  }  // Place two random tiles (2 or 4) at the beginning  srand(time(0));  int x1 = rand() % SIZE;  int y1 = rand() % SIZE;  int x2, y2;  do {  x2 = rand() % SIZE;  y2 = rand() % SIZE;  } while (x1 == x2 && y1 == y2);  board[x1][y1] = (rand() % 2 + 1) \* 2; // 2 or 4  board[x2][y2] = (rand() % 2 + 1) \* 2; // 2 or 4  score = 0;  }  void displayBoard() {  printf("\nScore: %d\n", score);  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  printf("%4d", board[i][j]);  }  printf("\n");  }  }  void moveLeft() {  // Move tiles to the left  for (int i = 0; i < SIZE; i++) {  int k = 0;  for (int j = 0; j < SIZE; j++) {  if (board[i][j] != 0) {  if (k != j) {  board[i][k] = board[i][j];  board[i][j] = 0;  }  k++;  }  }  // Merge tiles with the same value  for (int j = 0; j < SIZE - 1; j++) {  if (board[i][j] != 0 && board[i][j] == board[i][j + 1]) {  board[i][j] \*= 2;  board[i][j + 1] = 0;  score += board[i][j];  j++;  }  }  // Move tiles to the left again after merging  k = 0;  for (int j = 0; j < SIZE; j++) {  if (board[i][j] != 0) {  if (k != j) {  board[i][k] = board[i][j];  board[i][j] = 0;  }  k++;  }  }  }  }  int checkWin() {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  if (board[i][j] == WIN\_NUMBER)  return 1;  }  }  return 0;  }  int checkLose() {  for (int i = 0; i < SIZE; i++) {  for (int j = 0; j < SIZE; j++) {  if (board[i][j] == 0)  return 0;  if (i < SIZE - 1 && board[i][j] == board[i + 1][j])  return 0;  if (j < SIZE - 1 && board[i][j] == board[i][j + 1])  return 0;  }  }  return 1;  }  int main() {  initializeGame();  displayBoard();  char direction;  int validMove;  while (1) {  validMove = 0;  printf("\nEnter a direction (W - Up, A - Left, S - Down, D - Right): ");  scanf(" %c", &direction);  switch (direction) {  case 'W':  moveLeft();  validMove = 1;  break;  case 'A':  moveLeft();  validMove = 1;  break;  case 'S':  moveLeft();  validMove = 1;  break;  case 'D':  moveLeft();  validMove = 1;  break;  default:  break;  }  if (validMove) {  int x, y;  do {  x = rand() % SIZE;  y = rand() % SIZE;  } while (board[x][y] != 0);  board[x][y] = (rand() % 2 + 1) \* 2; // 2 or 4  displayBoard();  if (checkWin()) {  printf("\nYou win!\n");  break;  }  if (checkLose()) {  printf("\nGame over!\n");  break;  }  }  if (!validMove)  printf("\nInvalid move. Please try again.\n");  }  return 0;  } |