OS MINI PROJECT

[Company name] | [Company address]

**21F-9104 – 5D**

**Muhammad Saad Nadeem (INDIVIUAL GROUP)**

2023

|  |
| --- |
| OS MINI PROJECT |
| CODE: |
| #!/bin/bash  #########################################################################################################################################################  # Function to print the current state of the puzzle board  printPuzzle()  {  for ((i = 0; i < 5; i++)); do  for ((j = 0; j < 5; j++)); do  index=$((i \* 5 + j))  if [ ${board[$index]} -eq 0 ]; then  echo -n " |"  else  if [ ${board[$index]} -lt 10 ]; then  echo -n " "${board[$index]}"|"  else  echo -n " "${board[$index]}"|"  fi  fi  done  echo  done  echo  }  #########################################################################################################################################################  # Function to check if the current board configuration is the goal state  isGoal()  {  goalBoard=(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 0)  for ((i = 0; i < 25; i++)); do  if [ ${board[$i]} -ne ${goalBoard[$i]} ]; then  return 1  fi  done  return 0  }  #########################################################################################################################################################  # Function to perform the move and update the board configuration  move()  {    local temp=${board[$1]}  board[$1]=${board[$2]}  board[$2]=$temp  }  #########################################################################################################################################################  # Function to check if the move is valid  isValidMove()  {  local pos=$1  local emptyRow=$((emptyPos / 5)) #It calculates the row position of the empty cell by dividing emptyPos by 5.  local emptyCol=$((emptyPos % 5)) #It calculates the column position of the empty cell by taking the remainder of emptyPos divided by 5.  local rowDiff=$((emptyRow - pos / 5)) #It calculates the difference between the row of the empty cell and the row of the cell to be moved.  local colDiff=$((emptyCol - pos % 5)) #It calculates the difference between the column of the empty cell and the column of the cell to be moved.  local distance=$((rowDiff \* rowDiff + colDiff \* colDiff)) #It calculates the distance using the row and column differences.  #Whether pos is greater than or equal to 0 and less than 25.  #Whether the calculated distance is equal to 1 or 25.  if [ $pos -ge 0 ] && [ $pos -lt 25 ] && [ $distance -eq 1 ] || [ $distance -eq 25 ]; then  return 0  else  return 1  fi  }  #########################################################################################################################################################  # Function to play the game  playGame() {  board=(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 0 21 22 23 24) # Initial configuration, 0 represents the empty tile  goalBoard=(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 0) # Initial configuration, 0 represents the empty tile    echo "Initial Board Configuration: "  printPuzzle  echo  printGoal      while ! isGoal; do  read -p "Enter the number you want to move (0 for quit): " number  if [ $number -eq 0 ]; then  break  fi  emptyPos=-1  for ((i = 0; i < 25; i++)); do  if [ ${board[$i]} -eq 0 ]; then  emptyPos=$i  break  fi  done  numberPos=-1  for ((i = 0; i < 25; i++)); do  if [ ${board[$i]} -eq $number ]; then  numberPos=$i  break  fi  done  if isValidMove $numberPos; then  move $emptyPos $numberPos  printPuzzle  else  echo "Invalid move. Try again."  fi  done  if isGoal; then  echo "Congratulations! You solved the puzzle."  else  echo "Quitting the game."  fi  }  #########################################################################################################################################################  # Run the game  playGame  ######################################################################################################################################################### |
| **SCREENSHOT:** |
|  |