**CL2006-Operating System Lab**

Lab Instructor

**Ms. Sumaira Mustafa**

**Group Members:**

|  |  |
| --- | --- |
| **Name** | **Roll Number** |
| Rai Muhammad Saqlain | 21F-9255 |

**Section: 5E**

**24 -Puzzle Game**

Date of Submission

**Sunday, October 22, 2023**



**Department of Computer Science**

FAST – National University of Computer & Emerging Sciences

Chiniot-Faisalabad Campus.

**CODE:**

#!/bin/bash

#variables for colors

On\_White='\033[47m'

Black='\033[1;90m'

BBlue='\033[1;34m'

BBlack='\033[1;30m'

UYellow='\033[4;33m'

BYellow='\033[1;33m'

On\_Blue='\033[44m'

BIBlue='\033[1;94m'

BIYellow='\033[1;93m'

NC="\033[0m"

b\_size=5

game\_num=$((b\_size \* b\_size))

EMPTY\_SPACE=" "

declare -a board

declare -a g\_state\_game

# Function to initialize the game board with a random solvable state

init() {

local num=($(seq 1 $((game\_num - 1))))

num=($(shuf -e "${num[@]}"))

board=("${num[@]}")

board[24]="$EMPTY\_SPACE"

}

# This Function is used to print game board

Making\_board()

{

for ((i = 0; i < game\_num; i++)); do

printf "${BIYellow}${BIBlue}${On\_Blue}${BYellow}${UYellow}[%2s]${NC} " "${board[$i]}"

if ((i % b\_size == b\_size - 1)); then

echo

fi

done

}

# This Function is used to ensuring it is reachable from the initial state

goal()

{

local num=($(seq 1 $((game\_num - 1))) " ")

g\_state\_game=("${num[@]}")

if !Is\_Solvable; then

local temp=${g\_state\_game[$((game\_num - 1))]}

g\_state\_game[$((game\_num - 1))]=${g\_state\_game[$((game\_num - 2))]}

g\_state\_game[$((game\_num - 2))]=$temp

fi

}

# This Function is used to determine if the goal state is reachable from the initial state

Is\_Solvable()

{

local init\_b=$(Is\_reachable "${board[@]}")

local goal\_state=$(Is\_reachable "${g\_state\_game[@]}")

if ((b\_size % 2 == 1)); then

return $(((init\_b + goal\_state) % 2))

else

local initial\_blank\_row=$((b\_size - (init\_b / b\_size) - 1))

local goal\_blank\_row=$((b\_size - (goal\_state / b\_size) - 1))

local row\_diff=$((goal\_blank\_row - initial\_blank\_row))

return $(((init\_b+ goal\_state + row\_diff) % 2))

fi

}

# This function is help us to find the reachable state

Is\_reachable()

{

local -a arr=("$@")

local inversions=0

for ((i = 0; i < game\_num - 1; i++)); do

for ((j = i + 1; j < game\_num; j++)); do

if [[ ${arr[i]} -gt ${arr[j]} && ${arr[i]} != "$EMPTY\_SPACE" && ${arr[j]} != "$EMPTY\_SPACE" ]]; then

((inversions++))

fi

done

done

echo $inversions

}

# This Function is used to check if the current state matches the goal state

is\_goal()

{

local -a state=("${board[@]}")

for ((i = 0; i < game\_num; i++)); do

if [[ ${state[i]} != ${g\_state\_game[i]} ]]; then

return 1

fi

done

return 0

}

# This Function is used to get legal moves for the current state

legal\_moves()

{

local -a state=("${board[@]}")

local moves=()

for ((i = 0; i < game\_num; i++)); do

if [[ "${state[i]}" == "$EMPTY\_SPACE" ]]; then

local row=$((i / b\_size))

local col=$((i % b\_size))

if ((row > 0)); then

moves+=("U")

fi

if ((row < b\_size - 1)); then

moves+=("D")

fi

if ((col > 0)); then

moves+=("L")

fi

if ((col < b\_size - 1)); then

moves+=("R")

fi

fi

done

echo "${moves[@]}"

}

# This Function to make a move

make\_move()

{

local move\_val=$1

local EMPTY\_VAL

for ((i = 0; i < game\_num; i++)); do

if [[ "${board[$i]}" == "$EMPTY\_SPACE" ]]; then

EMPTY\_VAL=$i

break

fi

done

local row=$((EMPTY\_VAL / b\_size))

local col=$((EMPTY\_VAL % b\_size))

case $move\_val in

"U")

if ((row > 0)); then

local new\_val=$((EMPTY\_VAL - b\_size))

board[$EMPTY\_VAL]=${board[$new\_val]}

board[$new\_val]="$EMPTY\_SPACE"

fi

;;

"D")

if ((row < b\_size - 1)); then

local new\_val=$((EMPTY\_VAL + b\_size))

board[$EMPTY\_VAL]=${board[$new\_val]}

board[$new\_val]="$EMPTY\_SPACE"

fi

;;

"L")

if ((col > 0)); then

local new\_val=$((EMPTY\_VAL - 1))

board[$EMPTY\_VAL]=${board[$new\_val]}

board[$new\_val]="$EMPTY\_SPACE"

fi

;;

"R")

if ((col < b\_size - 1)); then

local new\_val=$((EMPTY\_VAL + 1))

board[$EMPTY\_VAL]=${board[$new\_val]}

board[$new\_val]="$EMPTY\_SPACE"

fi

;;

esac

}

# This Function to print the sequence of moves taken to reach the goal state

print\_path()

{

local -a state=("${board[@]}")

local path=()

while !is\_goal "${state[@]}"; do

for move in $(legal\_moves "${state[@]}"); do

local new\_state=("${state[@]}")

make\_move "$move"

if ! is\_goal "${new\_state[@]}"; then

state=("${new\_state[@]}")

path+=("$move")

break

fi

done

done

echo "Initial State of puzzle: ${board[@]}"

echo "Goal State of puzzle: ${g\_state\_game[@]}"

echo "Path of the game: ${path[@]}"

}

# This is the main code part

init

goal

while true; do

clear

echo -e " \U0001F60D ${On\_White}${Black} ${BBlack}HELLO WELCOME ${NC} ${NC}${NC}\U0001F60D "

echo -e " \U000270C ${On\_White}${Black} ${BBlack}24-Puzzle Game${NC} ${NC}${NC} \U000270C "

echo " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "

echo -e " \* ${BBlue}Please Enter U for going up \U0001F446${NC} \* "

echo -e " \* ${BBlue}Please Enter D for going Down \U0001F447${NC} \* "

echo -e " \* ${BBlue}Please Enter R for going Right \U0001F449${NC} \* "

echo -e " \* ${BBlue}Please Enter L for going Left \U0001F448${NC} \* "

echo -e " \* ${BBlue}Please Enter E or e for End Game \U0001F44D${NC} \* "

echo " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "

Making\_board

read -n 1 move

if [[ "$move" == "Q" || "$move" == "q" ]]; then

echo "Good bye!"

break

fi

make\_move "$move"

if is\_goal; then

clear

echo -e"${BBlue}Good job \U0001F44D you solve the puzzle \U0002764${NC}"

print\_board

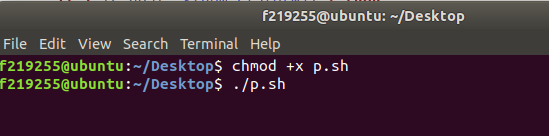
break

fi

done

**OUTPUT:**

**Commands:**

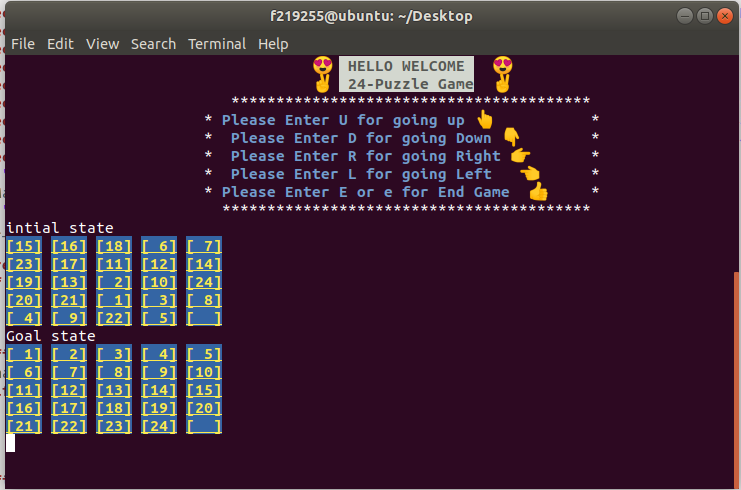


**Display:**

A screenshot of a computer

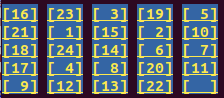
Description automatically generated

**Random Table Generate:**

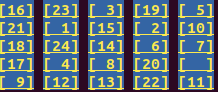
****

**Move UP :**

**Initial State :**

****

**Updated state:**

****

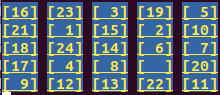
**Move LEFT:**

**Initial State :**

**A blue and yellow rectangular numbers

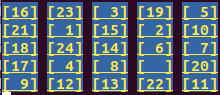
Description automatically generated with medium confidence**

**Updated state:**

****

**Move DOWN:**

**Initial State :**

****

**Updated state:**

**A blue rectangular sign with yellow numbers

Description automatically generated**

**Move RIGHT:**

**Initial State :**

**A blue rectangular sign with yellow numbers

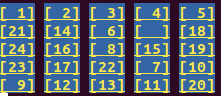
Description automatically generated**

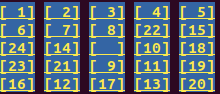
**Updated state:**

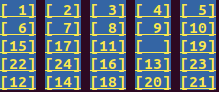
**A blue rectangular sign with yellow numbers

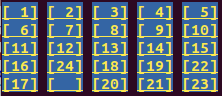
Description automatically generated**

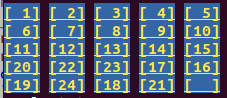
**Some Outputs of Table while solve the puzzle:**

****

****

****

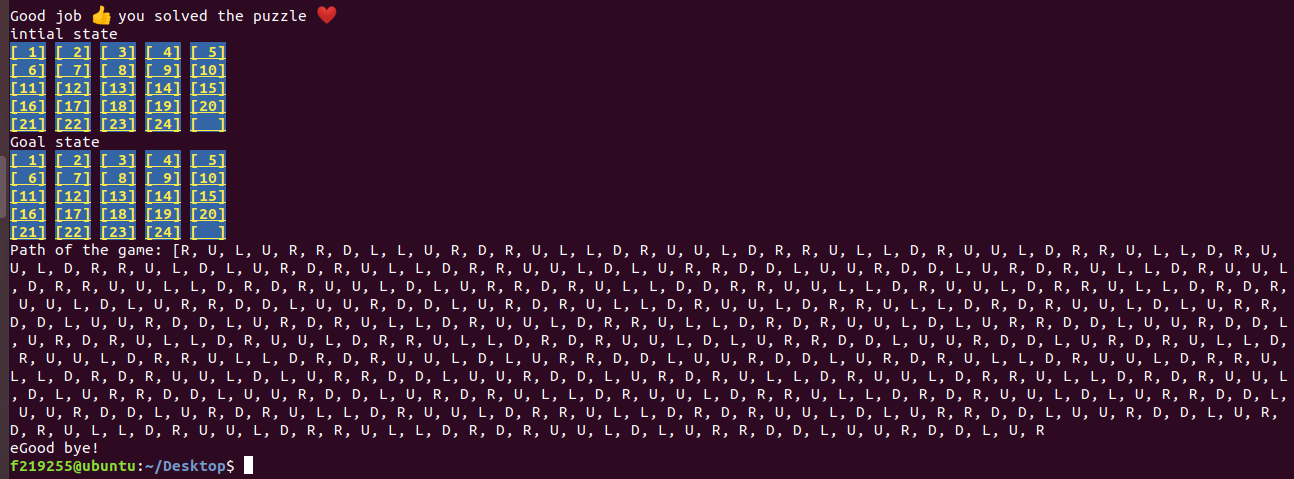
****

****

**A blue rectangular sign with yellow numbers

Description automatically generated**

**After Solve the puzzle:**

****