**OS Mini Project**

****

**Name:** Ahmad Imran

Muhammad Ammar

**Roll No:** 21F-9209

21F-9189

**Section:** BSCS-5D

# Code

#!/bin/bash

size=25

function goal\_stage {

for ((i=0; i<size; i++)); do

goal[$i]=$((i+1))

if [ "${goal[$i]}" -eq 25 ]; then

goal[$i]=0

fi

done

}

function is\_goal {

for ((i=0; i<size; i++)); do

if [ "${state[$i]}" -ne "${goal[$i]}" ]; then

return 1

fi

done

return 0

}

function print\_puzzle {

for ((i=0; i<size; i++)); do

echo -n "${arr[$i]} "

if [ $((i % 5)) -eq 4 ]; then

echo

fi

done

}

function initial\_stage {

for ((i=0; i<size; i++)); do

arr[$i]=$((i+1))

if [ "${arr[$i]}" -eq 25 ]; then

arr[$i]=0

fi

done

for ((i=size-1; i>0; i--)); do

j=$((RANDOM % (i+1)))

temp=${arr[$i]}

arr[$i]=${arr[$j]}

arr[$j]=$temp

done

}

function legal\_moves {

zeroPos=-1

for ((i=0; i<size; i++)); do

if [ "${arr[$i]}" -eq 0 ]; then

zeroPos=$i

fi

done

echo -n "Legal moves are:"

if [ $((zeroPos + 5)) -le $((size - 1)) ]; then

echo -n " u"

fi

if [ $((zeroPos - 1)) -ge 0 ] && [ $((zeroPos % 5)) -ne 0 ]; then

echo -n " l"

fi

if [ $((zeroPos + 1)) -le $((size - 1)) ] && [ $((zeroPos % 5)) -ne 4 ]; then

echo -n " r"

fi

if [ $((zeroPos - 5)) -ge 0 ]; then

echo " d"

else

echo

fi

}

function make\_move {

a=$1

zeroPos=-1

for ((i=0; i<size; i++)); do

if [ "${arr[$i]}" -eq 0 ]; then

zeroPos=$i

fi

done

case $a in

'd')

if [ $((zeroPos + 5)) -le $((size - 1)) ]; then

temp=${arr[$zeroPos]}

arr[$zeroPos]=${arr[$((zeroPos + 5))]}

arr[$((zeroPos + 5))]=$temp

else

echo "Up Move is not possible:"

fi

;;

'u')

if [ $((zeroPos - 5)) -ge 0 ]; then

temp=${arr[$zeroPos]}

arr[$zeroPos]=${arr[$((zeroPos - 5))]}

arr[$((zeroPos - 5))]=$temp

else

echo "Down Move is not possible:"

fi

;;

'l')

if [ $((zeroPos - 1)) -ge 0 ] && [ $((zeroPos % 5)) -ne 0 ]; then

temp=${arr[$zeroPos]}

arr[$zeroPos]=${arr[$((zeroPos - 1))]}

arr[$((zeroPos - 1))]=$temp

else

echo "Left Move is not possible:"

fi

;;

'r')

if [ $((zeroPos + 1)) -le $((size - 1)) ] && [ $((zeroPos % 5)) -ne 4 ]; then

temp=${arr[$zeroPos]}

arr[$zeroPos]=${arr[$((zeroPos + 1))]}

arr[$((zeroPos + 1))]=$temp

else

echo "Right Move is not possible:"

fi

;;

\*)

echo "Invalid input"

;;

esac

}

function count\_inversions {

local count=0

for ((i=0; i<size-1; i++)); do

for ((j=i+1; j<size; j++)); do

if [ "${arr[$i]}" -gt "${arr[$j]}" ] && [ "${arr[$i]}" -ne 0 ] && [ "${arr[$j]}" -ne 0 ]; then

count=$((count + 1))

fi

done

done

echo $count

}

function is\_solvable {

local goalInversions=$(count\_inversions goal)

local initStateInversions=$(count\_inversions state)

[ $((goalInversions % 2)) -eq $((initStateInversions % 2)) ]

}

# Main program starts here

goal\_stage

initial\_stage

echo "Initial stage of Array:"

print\_puzzle

echo "Goal stage of Array:"

print\_puzzle goal

if is\_solvable; then

echo "Solvable"

else

echo "Not solvable"

fi

echo "Minimum swaps for reached goal array: $(count\_inversions arr)"

while true; do

echo "Enter u for Up Move"

echo "Enter d for Down Move"

echo "Enter r for Right Move"

echo "Enter l for Left Move"

echo "Enter 0 for end the program:"

legal\_moves

read -r a

if [ "$a" = 'l' ] || [ "$a" = 'r' ] || [ "$a" = 'd' ] || [ "$a" = 'u' ]; then

make\_move "$a"

print\_puzzle

elif [ "$a" = '0' ]; then

break

else

echo "Invalid input"

fi

done

exit 0

# Output

