UNIX -WEEK-6

STATIC AND DYNAMIC LINKING.

Roll number:422139

Build\_And\_Run.sh.

#!/bin/bash

# Compile source files into object files

gcc -c -fPIC push\_pop.c NQueens.c

# Check if object files were created successfully

if [ $? -ne 0 ]; then

echo "Error: Failed to compile source files into object files"

exit 1

fi

# Create static library

ar rcs libFunctions.a push\_pop.o NQueens.o

# Check if static library was created successfully

if [ $? -ne 0 ]; then

echo "Error: Failed to create the static library"

exit 1

fi

# Create dynamic library

gcc -shared -o libFunctions.so push\_pop.o NQueens.o

# Check if dynamic library was created successfully

if [ $? -ne 0 ]; then

echo "Error: Failed to create the dynamic library"

exit 1

fi

# Set LD\_LIBRARY\_PATH to include current directory

export LD\_LIBRARY\_PATH=$(pwd):$LD\_LIBRARY\_PATH

# Compile main program with static library

gcc main.c -L. -lFunctions -o main\_static

# Check if main program with static library was compiled successfully

if [ $? -ne 0 ]; then

echo "Error: Failed to compile the main program with static library"

exit 1

fi

# Compile main program with dynamic library

gcc main.c -L. -lFunctions -o main\_dynamic

# Check if main program with dynamic library was compiled successfully

if [ $? -ne 0 ]; then

echo "Error: Failed to compile the main program with dynamic library"

exit 1

fi

# Execute main programs

echo "Executing main program with static library..."

./main\_static

echo "Executing main program with dynamic library..."

./main\_dynamic

echo "Compilation successful”

Functions.h.

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

#define MAX\_SIZE 10

struct Stack {

int items[MAX\_SIZE];

int top;

};

void initialize(struct Stack \*s);

int isFull(struct Stack \*s);

int isEmpty(struct Stack \*s);

void push(struct Stack \*s, int value);

int pop(struct Stack \*s);

void printBoard(int board[MAX\_SIZE][MAX\_SIZE], int n);

int isSafe(int board[MAX\_SIZE][MAX\_SIZE], int row, int col, int n);

void solveNQueens(int n);

#endif

//push\_pop.c

#include <stdio.h>

#include <stdlib.h>

#include "Functions.h"

void initialize(struct Stack \*s) {

s->top = -1;

}

int isFull(struct Stack \*s) {

return s->top == MAX\_SIZE - 1;

}

int isEmpty(struct Stack \*s) {

return s->top == -1;

}

void push(struct Stack \*s, int value) {

if (isFull(s)) {

printf("Stack Overflow\n");

return;

}

s->items[++(s->top)] = value;

}

int pop(struct Stack \*s) {

if (isEmpty(s)) {

printf("Stack Underflow\n");

exit(1);

}

return s->items[(s->top)--];

}

//NQueens.c

#include <stdbool.h>

#include <stdio.h>

#include "Functions.h"

void printBoard(int board[MAX\_SIZE][MAX\_SIZE], int n) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

printf("%d ", board[i][j]);

}

printf("\n");

}

printf("\n");

}

int isSafe(int board[MAX\_SIZE][MAX\_SIZE], int row, int col, int n) {

int i, j;

for (i = 0; i < col; i++) {

if (board[row][i]) {

return 0;

}

}

for (i = row, j = col; i >= 0 && j >= 0; i--, j--) {

if (board[i][j]) {

return 0;

}

}

for (i = row, j = col; j >= 0 && i < n; i++, j--) {

if (board[i][j]) {

return 0;

}

}

return 1;

}

void solveNQueens(int n) {

int board[MAX\_SIZE][MAX\_SIZE] = {0};

struct Stack positions;

initialize(&positions);

push(&positions, 0);

while (!isEmpty(&positions)) {

int col = pop(&positions);

if (col == n) {

printBoard(board, n);

return;

}

bool placed = false;

for (int i = 0; i < n; i++) {

if (isSafe(board, i, col, n)) {

board[i][col] = 1;

push(&positions, col + 1);

placed = true;

break;

}

}

if (!placed) {

if (!isEmpty(&positions)) {

int prev\_col = pop(&positions);

if (prev\_col == n - 1) {

return;

}

for (int i = 0; i < n; i++) {

if (board[i][prev\_col] == 1) {

board[i][prev\_col] = 0;

break;

}

}

push(&positions, prev\_col + 1);

}

}

}

printf("Solution does not exist\n");

}

//main.c

#include <stdio.h>

#include "Functions.h"

int main() {

int n;

printf("Enter no.of Queens : ");

scanf("%d",&n);

solveNQueens(n);

return 0;

}

