**Assignment 2**

**Problem:**

Given a stack, sort it using recursion. Use of any loop constructs like while, for etc. is not allowed. Use following functions on Stack S.

1. isEmpty(S) /\*Tests whether stack is empty or not\*/
2. push(S) /\* Adds new element to the stack\*/
3. pop(S) /\* Removes top element from the stack\*/
4. top(S) /\*Returns value of the top of the element\*/

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct Stack {

int data;

struct Stack\* next;

};

struct Stack\* newNode(int data) {

struct Stack\* stackNode = (struct Stack\*)malloc(sizeof(struct Stack));

stackNode->data = data;

stackNode->next = NULL;

return stackNode;

}

bool isEmpty(struct Stack\* root) {

return (root == NULL);

}

void push(struct Stack\*\* root, int data) {

struct Stack\* stackNode = newNode(data);

stackNode->next = (\*root);

(\*root) = stackNode;

}

int pop(struct Stack\*\* root) {

if (isEmpty(\*root)) {

printf("Stack is empty\n");

return -1;

}

struct Stack\* temp = \*root;

\*root = (\*root)->next;

int popped = temp->data;

free(temp);

return popped;

}

int top(struct Stack\* root) {

if (isEmpty(root)) {

printf("Stack is empty\n");

return -1;

}

return root->data;

}

void insertSorted(struct Stack\*\* root, int data) {

if (isEmpty(\*root) || data > top(\*root)) {

push(root, data);

return;

}

int temp = pop(root);

insertSorted(root, data);

push(root, temp);

}

void sortStack(struct Stack\*\* root) {

if (!isEmpty(\*root)) {

int temp = pop(root);

sortStack(root);

insertSorted(root, temp);

}

}

void printStack(struct Stack\* root) {

while (root != NULL) {

printf("%d ", top(root));

root = root->next;

}

printf("\n");

}

int main() {

struct Stack\* stack = NULL;

push(&stack, 30);

push(&stack, -5);

push(&stack, 18);

push(&stack, 14);

push(&stack, -3);

printf("Original stack: ");

printStack(stack);

sortStack(&stack);

printf("Sorted stack: ");

printStack(stack);

return 0;

}

**OUTPUT:**

Original stack: -3 14 18 -5 30

Sorted stack: -5 -3 14 18 30

**Assignment 3**

**Problem:**

Write a C program to implement a stack using linked list and accept some numeric values. Remove the number whose value is the minimum on the stack from the given variable.

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Stack {

struct Node\* top;

};

struct Node\* newNode(int data) {

struct Node\* node = (struct Node\*)malloc(sizeof(struct Node));

node->data = data;

node->next = NULL;

return node;

}

int isEmpty(struct Stack\* stack) {

return stack->top == NULL;

}

void push(struct Stack\* stack, int data) {

struct Node\* node = newNode(data);

node->next = stack->top;

stack->top = node;

}

int pop(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty.\n");

return INT\_MIN;

}

struct Node\* temp = stack->top;

int popped = temp->data;

stack->top = temp->next;

free(temp);

return popped;

}

int findMin(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty.\n");

return INT\_MIN;

}

int min = stack->top->data;

struct Node\* current = stack->top;

while (current != NULL) {

if (current->data < min) {

min = current->data;

}

current = current->next;

}

return min;

}

void removeMinValue(struct Stack\* stack) {

int min = findMin(stack);

struct Stack tempStack;

tempStack.top = NULL;

while (!isEmpty(stack)) {

int value = pop(stack);

if (value != min) {

push(&tempStack, value);

}

}

while (!isEmpty(&tempStack)) {

push(stack, pop(&tempStack));

}

}

void printStack(struct Stack\* stack) {

struct Node\* current = stack->top;

while (current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("NULL\n");

}

int main() {

struct Stack stack;

stack.top = NULL;

int n, value;

printf("Enter the number of elements to push onto the stack: ");

scanf("%d", &n);

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

printf("Enter value %d: ", i + 1);

scanf("%d", &value);

push(&stack, value);

}

printf("Original stack: ");

printStack(&stack);

removeMinValue(&stack);

printf("Stack after removing the minimum value: ");

printStack(&stack);

return 0;

}

**OUTPUT:**

Enter the number of elements to push onto the stack: 5

Enter value 1: 7

Enter value 2: 2

Enter value 3: 5

Enter value 4: 1

Enter value 5: 9

Original stack: 9 1 5 2 7 NULL

Stack after removing the minimum value: 9 5 2 7 NULL