#include <stdio.h>

#include <stdlib.h>

struct stack {

int data;

struct stack\* next;

};

void initStack(struct stack\*\* s) { \*s = NULL; }

int isEmpty(struct stack\* s)

{

if (s == NULL)

return 1;

return 0;

}

void push(struct stack\*\* s, int x)

{

struct stack\* p = (struct stack\*)malloc(sizeof(\*p));

if (p == NULL) {

fprintf(stderr, "Memory allocation failed.\n");

return;

}

p->data = x;

p->next = \*s;

\*s = p;

}

// Utility function to remove an item from stack

int pop(struct stack\*\* s)

{

int x;

struct stack\* temp;

x = (\*s)->data;

temp = \*s;

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

free(temp);

return x;

}

int top(struct stack\* s) { return (s->data); }

void sortedInsert(struct stack\*\* s, int x)

{

if (isEmpty(\*s) || x > top(\*s)) {

push(s, x);

return;

}

int temp = pop(s);

sortedInsert(s, x);

push(s, temp);

}

void sortStack(struct stack\*\* s)

{

if (!isEmpty(\*s)) {

int x = pop(s);

sortStack(s);

sortedInsert(s, x);

}

}

void printStack(struct stack\* s)

{

while (s) {

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

s = s->next;

}

printf("\n");

}

int main(void)

{

struct stack\* top;

initStack(&top);

push(&top, 5);

push(&top, -2);

push(&top, 9);

push(&top, -7);

push(&top, 3);

printf("Stack elements before sorting:\n");

printStack(top);

sortStack(&top);

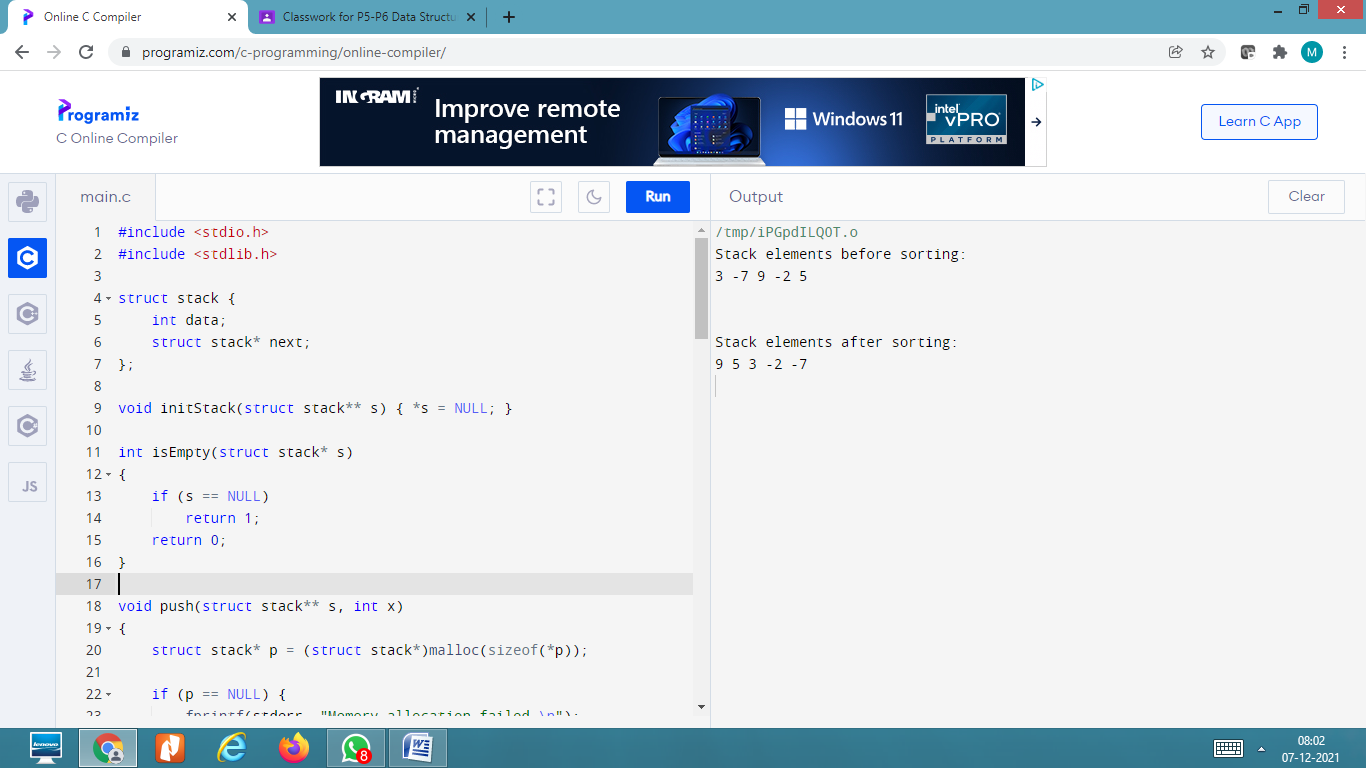
printf("\n\n");

printf("Stack elements after sorting:\n");

printStack(top);

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#define bool int

struct sNode {

char data;

struct sNode\* next;

};

void push(struct sNode\*\* top\_ref, int new\_data);

int pop(struct sNode\*\* top\_ref);

bool isMatchingPair(char character1, char character2)

{

if (character1 == '(' && character2 == ')')

return 1;

else if (character1 == '{' && character2 == '}')

return 1;

else if (character1 == '[' && character2 == ']')

return 1;

else

return 0;

}

bool areBracketsBalanced(char exp[])

{

int i = 0;

struct sNode\* stack = NULL;

while (exp[i])

{

if (exp[i] == '{' || exp[i] == '(' || exp[i] == '[')

push(&stack, exp[i]);

if (exp[i] == '}' || exp[i] == ')'

|| exp[i] == ']') {

if (stack == NULL)

return 0;

else if (!isMatchingPair(pop(&stack), exp[i]))

return 0;

}

i++;

}

if (stack == NULL)

return 1;

else

return 0;

}

// Driver code

int main()

{

char exp[100] = "{()}[]";

if (areBracketsBalanced(exp))

printf("true \n");

else

printf("false \n");

return 0;

}

void push(struct sNode\*\* top\_ref, int new\_data)

{

struct sNode\* new\_node

= (struct sNode\*)malloc(sizeof(struct sNode));

if (new\_node == NULL) {

printf("Stack overflow n");

getchar();

exit(0);

}

new\_node->data = new\_data;

new\_node->next = (\*top\_ref);

(\*top\_ref) = new\_node;

}

int pop(struct sNode\*\* top\_ref)

{

char res;

struct sNode\* top;

if (\*top\_ref == NULL) {

printf("Stack overflow n");

getchar();

exit(0);

}

else {

top = \*top\_ref;

res = top->data;

\*top\_ref = top->next;

free(top);

return res;

}

}

