**Batch: A3 Roll No.: 16010121051**

**Experiment / assignment / tutorial No.**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:**  Implementation of Stack applications. |

**Objective:** To implement applications of stack

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| 1 | Explain the different data structures used in problem solving |

**Books/ Journals/ Websites referred:**

1. *Fundamentals Of Data Structures In C –* Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications –* Jean Paul Tremblay,

Paul G. Sorenson

1. *Data Structures A Pseudo Approach with C –* Richard F. Gilberg & Behrouz A. Forouzan
2. [*https://www.cprogramming.com/tutorial/computersciencetheory/stack.html*](https://www.cprogramming.com/tutorial/computersciencetheory/stack.html)
3. [*https://www.geeksforgeeks.org/stack-data-structure-introduction-program/*](https://www.geeksforgeeks.org/stack-data-structure-introduction-program/)
4. [*https://www.thecrazyprogrammer.com/2013/12/c-program-for-array-representation-of-stack-push-pop-display.html*](https://www.thecrazyprogrammer.com/2013/12/c-program-for-array-representation-of-stack-push-pop-display.html)

**Assigned Stack application**:

 static, parenthesis match.

**Implementation:**

include <conio.h>

#include <stdio.h>

#include <string.h>

int top = -1;

int i;

char push(char\* stack,char c){

stack[++top] = c;

// printf("Element %c pushed\n",c);

}

char pop(char\* stack){

// printf("Element %c popped\n",stack[top]);

stack[top] = 0;

top--;

}

char peek(char\* stack){

// printf("%c\n",stack[top]);

return stack[top];

}

int isEmpty(char\* stack){

if(top == -1){

return 1;

}

else{

return 0;

}

}

void main(){

int size;

int flag = 0;

char a[100];

printf("Enter the equation: \n");

scanf("%s",a);

size=strlen(a);

char stack[size];

// printf("Length of string a = %zu \n",strlen(a));

printf("You entered %s \n",a);

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

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

printf("Scanned %c \n",a[i]);

if(peek(stack) == '(' && a[i] == ')'){

printf("popped %c \n",peek(stack));

pop(stack);

}

else{

if(a[i] == ')'){

flag = 1;

printf("WRONG");

break;

}

else{

push(stack,a[i]);

printf("pushed %c \n",a[i]);

}

}

}

}

printf("\n");

if(flag == 0){

if(isEmpty(stack) == 1){

printf("WOHOOOO CORRECT");

}

else{

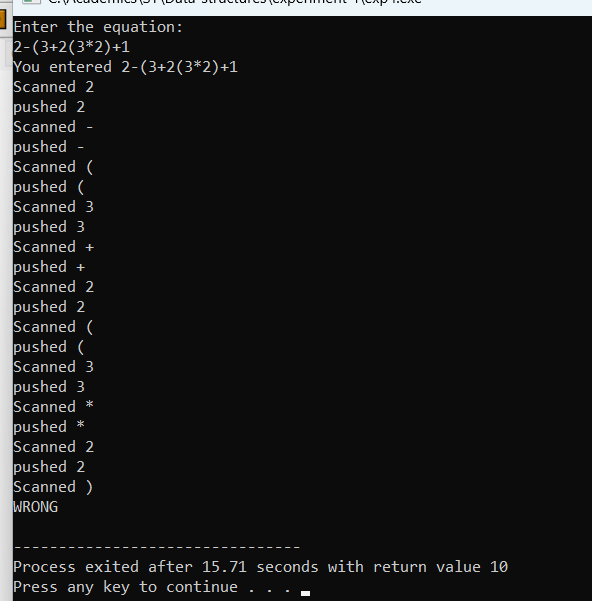
printf("OOPSSS WRONG");

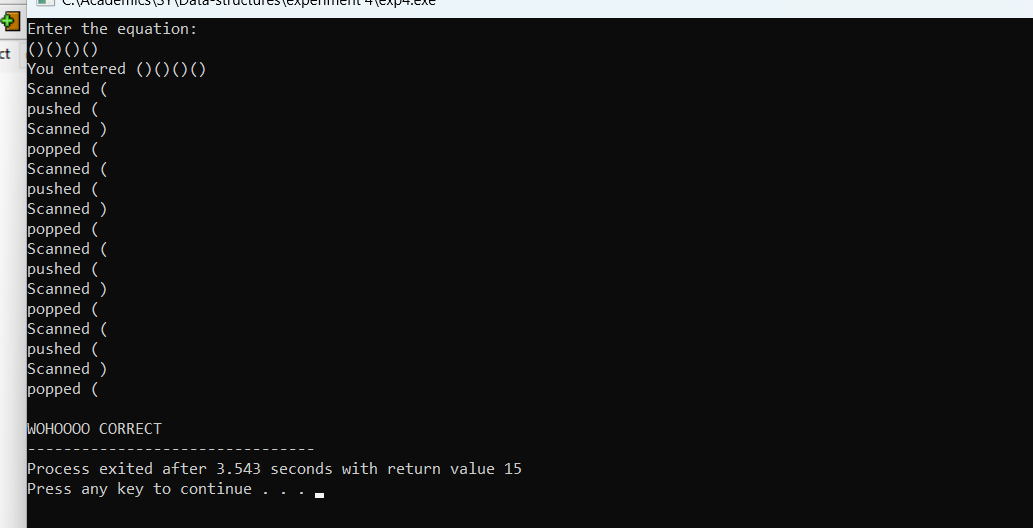
}

}

}

**Output Screenshots:**

****

****

**Conclusion:**

Implemented , parenthesis match using stack.