**DSA Assignment 3**

[**https://github.com/PrathamAsrani/DSA\_C/blob/master/assignment\_stack.c**](#_top)

#include <stdio.h>

#include <stdlib.h>

struct Stack

{

    char \*arr;

    int top, size;

};

int create(struct Stack \*s);

int isFull(struct Stack s);

int isEmpty(struct Stack s);

int push(struct Stack \*s);

int peek(struct Stack \*s);

int Balanced(struct Stack s);

int main()

{

    struct Stack stack;

    printf("Note : If you are providing input in Infix format please put a space between each character \n");

    create(&stack);

    push(&stack);

    if(Balanced(stack)){

        printf("Balanced\n");

    }else{

        printf("Un-balanced\n");

    }

    return 0;

}

int create(struct Stack \*s)

{

    s->top = -1;

    printf("Enter the size of stack : ");

    scanf("%d", &s->size);

    s->size = s->size \* 2;

    s->arr = (char \*)malloc(s->size \* sizeof(char));

}

int isFull(struct Stack s)

{

    if (s.top == s.size - 1)

    {

        return 1;

    }

    else

    {

        return 0;

    }

}

int isEmpty(struct Stack s)

{

    if (s.top = -1)

    {

        return 1;

    }

    else

    {

        return 0;

    }

}

int push(struct Stack \*s)

{

    if (isFull(\*s))

    {

        printf("Stack is full\n");

    }

    else

    {

        char character;

        printf("Enter characters : \n");

        for (int i = 0; i < s->size; i++)

        {

            s->top++;

            scanf("%c", &character);

            \*(s->arr + s->top) = character;

            // printf("%c ", s->arr[i]);

        }

    }

}

int pop(struct Stack \*s){

    if(isEmpty(\*s)){

        printf("Stack is empty\n");

    }else{

        int top = s->arr[s->top];

        s->top--;

        return top;

    }

}

int peek(struct Stack \*s)

{

    if (isEmpty(\*s))

    {

        printf("Stack is empty\n");

    }

    else

    {

        return s->arr[s->top];

    }

}

int Balanced(struct Stack s){

    int count\_lp = 0, count\_rp = 0;

    for(int i = 0; i < s.size; i++){

        if(s.arr[i] == '('){

            count\_lp++;

        }else if(s.arr[i] == ')'){

            count\_rp++;

        }

    }

    if(count\_lp == count\_rp){

        return 1;

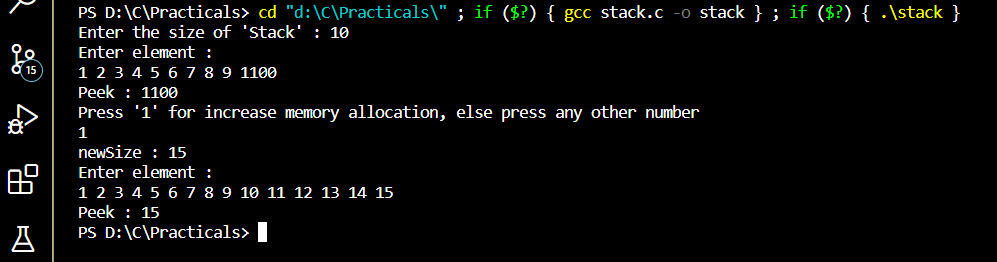
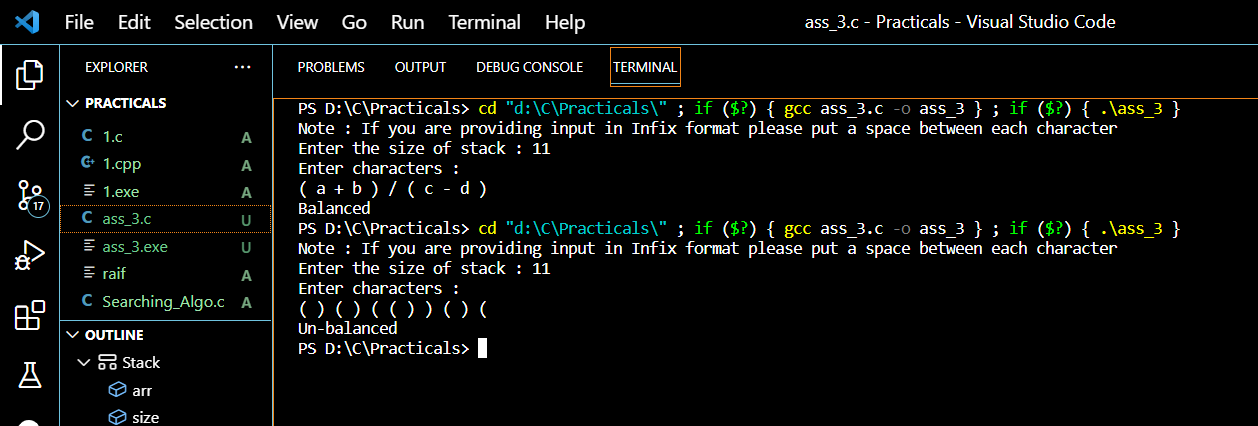
    }else{

        return 0;

    }

}

**Output:**

****

**Result : Hence we used stack to compare the infix expression with parenthesis is balanced or not.**