Nisarg Wath
642403014

Assignment 3

Expression

Header.h

```
typedef struct Stack{
    char * data;
    int size;
    int top;

}Stack;


void init(Stack *s, int size);
void push(Stack * s, char value);
char pop(Stack *s);
int check(char exp[]);
```

Logic.c

```
#include <stdio.h>
#include <stdlib.h>
#include "header.h"



void init(Stack *s, int size) {
    s->data = (char *)malloc(size * sizeof(char));  // Allocate
memory for 'size' characters
    s->top = -1;
    s->size = size;
}


void push(Stack *s, char value) {
```

```
    if (s->top == s->size - 1) {
        printf("Stack Overflow!!\n");
        return;
    }
    s->data[++(s->top)] = value;
}


char pop(Stack *s) {
    if (s->top == -1) {
        printf("Stack is Empty!\n");
        return '\0';
    }
    return s->data[s->top--];
}


char peek(Stack *s) {
    if (s->top == -1) {
        return '\0';
    }
    return s->data[s->top];
}


int check(char exp[]) {
    Stack *s = (Stack *)malloc(sizeof(Stack));
    init(s, 101);
    int i = 0;

    while (exp[i] != '\0') {
        if (exp[i] == '(' || exp[i] == '{' || exp[i] == '[') {
            push(s, exp[i]);
        } else if (exp[i] == ')' || exp[i] == ']' || exp[i] == '}'
){
            if (peek(s) == '('  || peek(s) == '[' || peek(s) ==
'{') {
                pop(s);
            } else {

                free(s->data); // Clean up memory
                free(s);
                return 0;
            }
        }
        i++;
    }


    int result = (s->top == -1) ? 1 : 0;
```

```
    free(s->data);
    free(s);

    return result;
}
```

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "header.h"



int main() {
    char expression[] =  "[()]{}{[()()]()}";

    if (check(expression)) {
        printf("True\n");
    } else {
        printf("False\n");
    }

    return 0;
}
```