```
#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;
}


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


    if (areBracketsBalanced(exp))
        printf("Balanced \n");
    else
        printf("Not Balanced \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;
    }
}
```