

Program to check for balanced brackets in an expression using stack!

Ex:

Input: exp = "[ ( ) ] { [ ( ) ] }

Output: Balanced

### Algorithm

\* Declare a character stack S

\* Now traverse the expression string exp

① If the current character is a starting bracket then push it to stack

② If the current character is a closing bracket then pop from stack and if the popped character is the matching starting bracket then fine else brackets are not balanced.

\* After completing traversal, if there is some starting bracket left in stack then "not balanced"

### Program

```
#include <stdio.h>
```

```
#define MAX 100
```

```
bool isMatchingPair(char a, char b){
```

```
    return (a == '(' && b == ')') ||
```

```
           (a == '[' && b == ']') ||
```

```
           (a == '{' && b == '}');
```

```
}
```

```

bool areBracketsBalanced(char exp[])
{
    char stack[MAX];
    int top = -1;
    for (int i = 0; exp[i]; i++)
    {
        if (exp[i] == '(' || exp[i] == '[' || exp[i] == '{')
        {
            if (top == MAX - 1) return false;
            stack[++top] = exp[i]; // Push
        }
        else if (exp[i] == ')' || exp[i] == '}' || exp[i] == ']')
        {
            if (top == -1 || !isMatchingPair(stack[top--], exp[i]))
                return false;
        }
    }
    return top == -1;
}

```

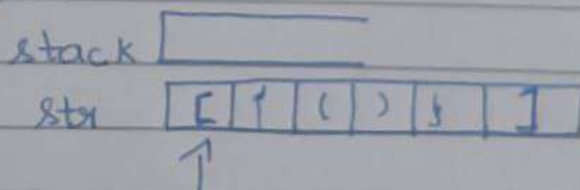
```

int main()
{
    char exp[] = "{}()[]";
    printf("are Brackets Balanced (exp)? \"Balanced\n\" :  
        \"Not Balanced\n\");
    return 0;
}

```

Iteration :

Initially :



Step 1: <sup>stack</sup>

E	
---	--

str

[	{	(	)	}	]
---	---	---	---	---	---

  
↑

opening bracket. Push into stack

Step 2:

stack 

E	{		
---	---	--	--

str 

[	{	(	)	}	]
---	---	---	---	---	---

  
↑

opening bracket. Push into stack

Step 3:

stack 

E	{	(	
---	---	---	--

str 

[	{	(	)	}	]
---	---	---	---	---	---

  
↑

Closing bracket. Check top of the stack is same kind or not

Step 4: stack 

E	{		
---	---	--	--

str 

[	{	(	)	}	]
---	---	---	---	---	---

  
↑

Step 5: stack 

E			
---	--	--	--

str 

[	{	(	)	}	]
---	---	---	---	---	---

  
↑

∴ Stack = top-1 hence balanced