Started on	Wednesday, 10 April 2024, 10:19 AM
State	Finished
Completed on	Wednesday, 10 April 2024, 11:52 AM
Time taken	1 hour 32 mins
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	TAMILARASI R 2022-CSD-A

Question **1**

Correct

Mark 1.00 out of 1.00

Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

For example:

Input	Result
break	break is a keyword
IF	IF is not a keyword

Answer: (penalty regime: 0 %)

```
key=[]
n=str(input())
key=['break','case','continue','default','defer','else','for','func'
if n in key:
    print("%s is a keyword"%(n))
else:
    print("%s is not a keyword"%(n))
```

	Input	Expected	Got	
✔	break	break is a keyword	break is a keyword	✔

	Input	Expected	Got	
✔	IF	IF is not a keyword	IF is not a keyword	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Find if a String2 is substring of String1. If it is, return the index of the first occurrence. else return -1.

Sample Input 1

thistest123string

123

Sample Output 1

8

Answer: (penalty regime: 0 %)

```
str1=str(input())
str2=str(input())
print(str1.find(str2))
```

	Input	Expected	Got	
✔	thistest123string 123	8	8	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

For example:

Input	Result
break	break is a keyword
IF	IF is not a keyword

Answer: (penalty regime: 0 %)

```
key=[]
n=str(input())
key=['break','case','continue','default','defer','else','for','func'
if n in key:
    print("%s is a keyword"%(n))
else:
    print("%s is not a keyword"%(n))
```

	Input	Expected	Got	
✔	break	break is a keyword	break is a keyword	✔

	Input	Expected	Got	
✔	IF	IF is not a keyword	IF is not a keyword	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

```
Input:
A man, a plan, a canal: Panama

Output:
1
```

Example 2:

```
Input:
race a car

Output:
0
```

Constraints:

- `s` consists only of printable ASCII characters.

Answer: (penalty regime: 0 %)

```
x = input()
x = x.lower()
x = ''.join(char for char in x if char.isalpha())
y = x[::-1]
if(y==x):
        print("1")
else:
    print("0")
```

	Input	Expected	Got	
✔	A man, a plan, a canal: Panama	1	1	✔
✔	race a car	0	0	✔

Passed all tests! ✔

Correct

Question **5**

Correct

Mark 1.00 out of 1.00

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

Constraints:

1 <= s.length <= 10^4

s consists of parentheses only '()[]{}'.

For example:

Input	Result
()	true
()[]{}	true
(]	false

Answer: (penalty regime: 0 %)

```
def is_balanced_brackets(s):
    stack = []
    brackets_map = {')': '(', '}': '{', ']': '['}
    for char in s:
        if char in brackets_map.values():
            stack.append(char)
        elif char in brackets_map.keys():
            if not stack or stack[-1] != brackets_map[char]:
                return False
            stack.pop()
    return not stack
n=input()
if is_balanced_brackets(n):
    print("true")
else:
     print("false")
```

	Input	Expected	Got	
✔	()	true	true	✔
✔	()[]{}	true	true	✔
✔	(]	false	false	✔

Passed all tests! ✔

Correct