

Lesson:



Recursion – 3



Pre-Requisites

- Recursion basics
- Working rules of recursive functions

List of Concepts Involved

- Recursion on strings

Topic : Recursion on strings

The method doesn't differ much from the recursion used in arrays.
Let's see some questions

Example 1 Remove all the occurrences of 'a' from string s = "abcax".

Input s="abcax" ch='a'

Output bcx

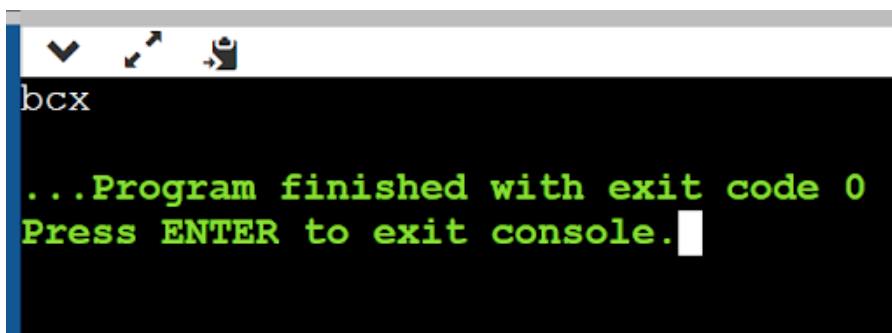
Explanation: We have been given the string "abcax", we have to remove all occurrences of char a from the string "abcax", and the string contains 2 'a'. So after removing the occurrences of a from string we will get "bcx" as output.

Approach:

- We have made a recursive function removeChar whose return type is string which takes string input and char ch whose occurrences is to be removed.
- Check for the base case i.e if string is empty, then return empty string.
- Check if the first char of string matches with char ch, if matches then call function removeChar with substring starting from index 1 (or in other words if first char of string matches with char ch then remove that char from string). If it does not match, add the first char of the string in the result and call function removeChar for the leftover string (substring starting with index 1).

Code

<https://pastebin.com/6qr5XgHr>



```
bcx

...Program finished with exit code 0
Press ENTER to exit console.
```

Example 2 Write a program to check whether a given number is palindrome or not.

Input num = 1234321

Output Yes

Explanation: This number remains the same when digits are reversed as we know that a palindrome number is a number that remains the same when digits are reversed.

Approach:

- We will check whether a given number is palindrome or not without converting it into string using the concept of recursion.
- We will create a copy of num and store it in the temp variable.
- We have created a function checkUtil to check if a num is palindrome or not by reading num from left to right and temp from right to left.
- Base case condition is here if the num is between 0 and 9 (i.e single digit) and if it's true then we will compare the num here with the last digit of temp and return the result.
- If num is not a single digit, we will check for num/10 and call the function checkUtil(num/10,temp), This is so, because it is not possible to read a number from left to right if it is an integer. If it returns false, we will return false and stop the process but if it returns true i.e. (num/10 is palindrome) then we will check for if num%10 contains i'th digit from beginning, then (*temp)%10 contains i'th digit from end and return the result.
- Note that all recursive calls have a separate copy of num, but they all share same copy of *temp. We divide num while moving up the recursion stack and we are reducing temp by 10 i.e (*temp/10).
- That is how we are comparing (i-1)th digit from end in temp and comparing it with (i-1)th digit from the beginning of nums.

Explaining one example how the code is actually working :

Let's take an example for better understanding.

Let num = 12321

Then temp = 12321

During first iteration of the code :

Num = 12321 Since num does not lie between 0 and 9 therefore this base case will be skipped at once.

Now it will come to the 11th line and another call will be made to the same function with parameters num/10 and temp.

Now the same function will execute for num = 1232 and temp = 12321.

Again the same case, num does not lie between 0 and 9, henceforth the base case will not be executed.

A new call will be sent that is this time num = 123 and temp = 12321

Again the base case will not be satisfied and function will be called again for num = 12 and temp = 12321

Again the function will be called for num = 1 and temp = 12321

This time because num lies between 0 and 9 that means we are entering the base case condition.

Now the thing is we have already discussed that we will compare num from left to right and temp from right to left. Here we got the leftmost digit of num that means we need to extract the rightmost digit of temp. That is very simple to extract. We can just write temp%10 that gives us 1 and num is also 1. Since both num and temp are equal that means, at least one digit is forming a palindrome.

Now since this call has returned true now we will go to other pending calls that are 12, 123, 1232 and 12321.

Now the next call would be for num = 12. Now if we are comparing num with temp%10 then temp%10 is 1 then that would be wrong, since that 1 is satisfied with 1 of num. To compare the second digit of num we want the second last digit from temp. To do that we need to reduce temp by 1 digit. That is possible only when we divide temp by 10. So now we do temp/10 and temp becomes 1232.

To compare, we know that first digit of num is already satisfied with last digit of temp. So out of 12 we only need to compare for 2 and 1 is already a part of palindrome. So in order to retrieve the last digit of num again we have to do num%10 so we get 2 and this needs to be compared with last digit of temp so we did temp%10 = 2. Again both the digits are matching therefore complete 12 is a part of palindrome and now we need to look for remaining function calls that are of 123, 1232 and 12321.

Again temp should be reduced because now we are going to compare third digit from num and third last digit from temp. So temp/10 = 123 and we already know that "12" are a part of palindrome so we will only compare that 3. So we did num%10 and temp%10. Again both are equal. We went to remaining calls that are of 1232 and 12321.

We did temp/10 = 12.

Again we know "123" is already a palindrome therefore we did num%10 = 2 and temp%10 = 2.

Now since this is also equal, it shows that 1232 is already a palindrome. Just a last check and we can conclude the results out.

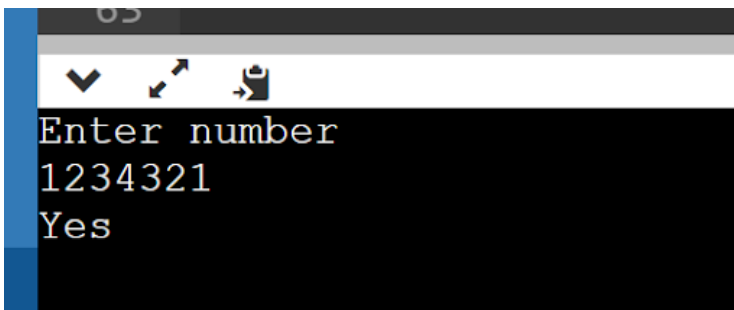
Last call is for 12321. Temp/10 = 1. "1232" is already settled as palindrome.

Now num%10 = 1 and temp%10 = 1 therefore this will also return true.

Hence it is concluded that the complete number 12321 is a palindrome.

Code link

<https://pastebin.com/42xfks85>



```

Enter number
1234321
Yes
  
```

Upcoming Class Teasers

- Problems based on recursion.