L7 : Advanced Recursion Practice Questions

1-Tut : **Replace Character Recursively**

Send Feedback

Given an input string S and two characters c1 and c2, you need to replace every occurrence of character c1 with character c2 in the given string.

Do this recursively.

Input Format :

The first line of input will contain an integer T, which will denote the value of number of test cases. In the following lines, T test cases will be written.
The first line of each test case will contain a string S.
The following line of each test case will contain two space separated characters, c1 and c2, respectively.

Output Format :

For each test case, the first and only line of output contains the updated string S.

Constraints :

1 <= T <= 100
1 <= Length of String S <= 100

Sample Input :

```
1
abacd
a x
```

Sample Output :

```
xbxcd
```

```
#include<bits/stdc++.h>
using namespace std;
void replacechar(char s[], char c1, char c2)
{
     if(s[0] == '\0'){
         return;
     }
    if(s[0] == c1){
        s[0] = c2;
    }
    replacechar(s+1,c1,c2);

}
int main(){

    // write your code here
    int T; cin >> T;
    while(T--)
    {
        char s[100]; cin >> s;
        char c1,c2;
        cin >> c1 >> c2;
        replacechar(s,c1,c2);
        cout << s << endl;

    }
    return 0;
}
```

2-Tut : Remove Duplicates Recursively

Send Feedback

Given a string S, remove consecutive duplicates from it recursively.

Input Format :

First line of input will contain T number of test cases

Next T line will contain the string S

Output Format :

For every test case print the answer in a separate line

Constraints :

1 <= T <= 10

1 <= |S| <= 10^4

where |S| represents the length of string

Sample Input 1 :

```
1
aabccba
```

Sample Output 1 :

```
abcba
```

Sample Input 2 :

```
1
xxxyyyzwwzzz
```

Sample Output 2 :

```
xyzwz
```

```
#include<bits/stdc++.h>
using namespace std;
void removeconsduplicate(char s[]){
    if (s[0] == '\0'){
        return;
    }
    if(s[0] == s[1]){
        int i=0;
        while(s[i] != '\0')
        {
            s[i]=s[i+1];
            i++;
        }
        removeconsduplicate(s);
    }
     removeconsduplicate(s+1);

}
int main(){

    // write your code here
    int T; cin >> T;
    while(T--)
    {
        char s[10000];
        cin >> s;
        removeconsduplicate(s);
        cout << s << endl;

    }
    return 0;
}
```

3-Tut : **Merge Sort Code**

Send Feedback

Sort an array A using Merge Sort.

Change in the input array itself. So no need to return or print anything.

Input format :

First line of input will contain T number of test cases
First line of every input will contain a single integer N size of the input array.
second line of each input will contain N space-separated integers.

Output format :

For every test case print, array elements in increasing order (separated by space) in a separate line.

Constraints :

1 <= T <= 10
1 <= n <= 10^5

Sample Input 1 :

```
1
6
2 6 8 5 4 3
```

Sample Output 1 :

```
2 3 4 5 6 8
```

Sample Input 2 :

```
1
5
2 1 5 2 3
```

Sample Output 2 :

```
1 2 2 3 5
```

```
#include<bits/stdc++.h>
using namespace std;
void merge(int input[], int si, int ei){

    int mid = (si + ei)/2;
    int size = ei -si +1;
    int arr[size];
    int i = si, j = mid +1, k = 0;

    while( i <= mid && j <= ei){

        if(input[i] <= input[j]){
           arr[k++] = input[i++];
        }
        else if(input[j] < input[i]){
            arr[k++] = input[j++];
        }
    }

    while(i <= mid){
        arr[k++] = input[i++];
    }
    while(j <= ei){
        arr[k++] = input[j++];
    }
    i = si;
    j = 0;
    while(j < k){
        input[i++] = arr[j++];
    }
}



void mergesort(int arr[],int si,int ei){
    if(si >= ei){
        return;
    }
    int mid = (si+ei)/2;
    mergesort(arr,si,mid);
    mergesort(arr,mid+1,ei);

    merge(arr,si,ei);

}
int main(){

    // write your code here
    int T; cin >> T;

    while(T--){
        int N; cin >> N;
        int arr[N];

        for(int i = 0; i < N; i++){
            cin >> arr[i];
        }
        mergesort(arr,0,N-1);
        for(int i = 0; i < N; i++){
            cout << arr[i] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

4-Tut : Quick Sort Code

Send Feedback

Sort an array A using Quick Sort.

Change in the input array itself. So no need to return or print anything.

Input format :

First line will contain T number of test cases and each test case will consist of two lines.

Line 1 : Integer n i.e. Array size

Line 2 : Array elements (separated by space)

Output format :

for every test case print array elements in increasing order (separated by space) in a new line.

Constraints :

1 <= T <= 10

1 <= n <= 10^5

0 <= arr[i] <= 10^9

Sample Input 1 :

```
1
6
2 6 8 5 4 3
```

Sample Output 1 :

```
2 3 4 5 6 8
```

Sample Input 2 :

```
1
5
1 5 2 7 3
```

Sample Output 2 :

```
1 2 3 5 7
```

```
#include<bits/stdc++.h>
using namespace std;
int partition(int arr[],int si,int ei){

    int pivot = arr[si];
    int countsmaller = 0;
    int i = si+1;

    while(i <= ei){
        if(arr[i] <= pivot){
            countsmaller++;
        }
        i++;
    }

    int pi = si+countsmaller;
    arr[si] = arr[pi];
    arr[pi] = pivot;
    i = si;
    int j  = ei;

    while(i < pi && j > pi){
        if(arr[i] <= pivot){
            i++;
        }
        else if(arr[j] > pivot){
            j--;
        }else{
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
            j--;
        }

    }
    return pi;
}

void quicksort(int arr[], int si, int ei){
    if(si >= ei){
        return;
    }

    int pi = partition(arr,si,ei);
    quicksort(arr,si,pi-1);
    quicksort(arr,pi+1,ei);
}

```

```
int main(){

    // write your code here
    int T; cin >> T;
    while(T--){
        int N;cin >> N;
        int arr[N];
        for(int i = 0; i < N; i++){
            cin >> arr[i];
        }
        quicksort(arr,0,N-1);
        for(int i = 0; i < N; i++){
            cout << arr[i] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

5-Tut : **Return Keypad Code**

Send Feedback

Given an integer n, using phone keypad find out all the possible strings that can be made using digits of input n.

The numbers and their corresponding codes are given below:

```
0: ""
1: ""
2: "abc"
3: "def"
4: "ghi"
5: "jkl"
6: "mno"
7: "pqrs"
8: "tuv"
9: "wxyz"
```

Return empty string for numbers 0 and 1.

Note:

```
1. The order of strings are not important.
2. The input number will have digits between: [2, 9].
```

Input Format :

First line of input will contain T number of test cases.
Each input consists of a single line containing an integer n.

Output Format :

For each test case, print all possible strings in different lines.

Constraints :

1 <= T <= 100

```
1 <= n <= 10^6
```

Sample Input:

```
1
23
```

Sample Output:

```
ad
ae
af
bd
be
bf
cd
ce

cf
```

```
#include<bits/stdc++.h>
using namespace std;
string options[] = {" "," ", "abc", "def", "ghi", "jkl", "mno","pqrs","tuv","wxyz"};
int keypad(int num, string output[]){
    /* Insert all the possible combinations of the integer number into the output string array. You do not need to
    print anything, just return the number of strings inserted into the array.
    */
    if(num == 0 || num == 1){
        return 1;
    }

    int ld = num % 10;
  //  num = num/10;
    int smallanssize = keypad((num / 10 ), output);

    int k = 0;
    for(int i = 0; i < options[ld].size(); i++){

        for(int j = 0; j < smallanssize; j++){
            output[k] = output[j];
            k++;
        }
    }

    k = 0;
    for(int i = 0; i < options[ld].size(); i++){

        for(int j = 0; j < smallanssize; j++){
            output[k] = output[k] + options[ld][i];
            k++;
        }
    }

    return ((options[ld].size()) * smallanssize);
}

int main(){

    // write your code here
    int T; cin >> T;
    while(T--){
    int n; cin >> n;

    string output[10000];
    int count = keypad(n, output);
    for(int i = 0; i < count && i < 10000; i++){
        cout << output[i] << endl;
    }
    }
    return 0;
}
```

6-Tut : **Print Keypad Combinations Code**

Send Feedback

Given an integer n, using phone keypad find out and print all the possible strings that can be made using digits of input n.

Note : The order of strings are not important. Just print different strings in new lines.

Input Format :

Integer n

Output Format :

All possible strings in different lines

Constraints :

1 <= n <= 10^6

Sample Input:

```
23
```

Sample Output:

```
ad
ae
af
bd
be
bf
cd
ce

cf
```

```
#include <iostream>
#include <string>
using namespace std;
string options[] = {" "," ", "abc", "def", "ghi", "jkl", "mno","pqrs","tuv","wxyz"};
void printKeypadhelper(int num , string output){

    if(num == 0){
        cout << output << endl;
        return;
    }
    int ld = num%10;
    int smallnum = num / 10;
    string option = options[ld];
    for(int i = 0; i < option.size(); i++){

        printKeypadhelper(smallnum, option[i] + output);

    }
}


void printKeypad(int num){
    /*
    Given an integer number print all the possible combinations of the keypad. You do not need to
return anything just print them.
    */
    string output = "";
    printKeypadhelper(num,output);

}
```