**Swapping Characters**

Attempted by: **271**

/

Accuracy: **37%**

/

Maximum Score: **20**

/

5 Votes

Tag(s):

Easy

**PROBLEM**

**EDITORIAL**

**MY SUBMISSIONS**

**ANALYTICS**

Your are given a string of length **N**. You have to follow some rounds of swapping in the string until the below explained condition is reached.

In each round the, you have to swap the **ith and (i+1)th character** , then **(i+2)th and (i+3)th** character and continue till the end of string. In each round , a character from the left will be locked i.e not available for swapping after this round , thus after some rounds of swapping , all the characters will be locked at their final position to stop any more rounds of swapping .

See the sample explanation for more clarity.

**Input**

The first line of input contains a **T**, the number of test case. The first line of each Test case will contain an Integer **N** , denoting the length of the string. The second line of each Test case will contain a string of length **N**.

**Output**

For each and every test case, output the string after all positions are locked.

**Constraints**

1. 0<**T**<100
2. 0<**N**<10000

**SAMPLE INPUT**

2

6

abcdef

3

abc

**SAMPLE OUTPUT**

bdfeca

bca

**Explanation**

For the first test case , the given string is "abcdef" and all the characters are unlocked.

1. After First **round** of swap , the string becomes "badcfe" . Now the leftmost unlocked character is locked i.e the string is now "b\_adcfe".
2. After Second **round** of swap , the string becomes "b\_dafce". Now the leftmost unlocked character is locked i.e string is now "bd\_afce".
3. After Third **round** of swap , the string becomes "bdfaec" . Now the leftmost unlocked character is locked i.e the string is now "bdf\_aec".
4. After Fourth **round** of swap , the string becomes "bdfeac" . Now the leftmost unlocked character is locked i.e the string is now "bdfe\_ac"..
5. After Fifth **round** of swap , the string becomes "bdfeca" . Now the leftmost unlocked character is locked i.e the string is now "bdfec\_a".
6. In the sixth round no more swap occur as there is only single unlocked character is left.

\*\* ('-' is the partition , where all the characters to its left are locked characters and all the characters to it right are unlocked characters).\*\*

Therefore , we print the string "bdfeca".

**Time Limit:**0.15 sec(s) for each input file.

**Memory Limit:**256 MB

**Source Limit:**1024 KB

**Marking Scheme:**Marks are awarded when all the testcases pass.

**Allowed Languages:**C, C++, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java, Java 8, JavaScript(Rhino), JavaScript(Node.js), Lisp, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, R(RScript), Racket, Ruby, Rust, Scala, Scala 2.11.8, Swift, Visual Basic

<https://www.hackerearth.com/practice/basic-programming/implementation/basics-of-implementation/practice-problems/algorithm/swapping-string-4/>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

int t = int.Parse(Console.ReadLine());

while (t-- > 0)

{

int n = int.Parse(Console.ReadLine());

string s = Console.ReadLine();

char[] ch = s.ToCharArray();

char[] res = new char[ch.Length];

int izq = 0, der = res.Length - 1;

int i = 0;

for (i = 0; i + 1 < ch.Length; i += 2)

{

res[der] = ch[i];

res[izq] = ch[i + 1];

izq++;

der--;

}

if (izq == der)

{

res[izq] = ch[i];

}

Console.WriteLine(new string(res));

}

Console.ReadLine();

}

}

}