Problem: RICHIE-RICH

Palindromes are strings that read the same from the left or right, for example *madam* or *0110*.

You will be given a string representation of a number and a maximum number of changes you can make. Alter the string, one digit at a time, to create the string representation of the largest number possible given the limit to the number of changes. The length of the string may not be altered, so you must consider 's left of all higher digits in your tests. For example  is valid,  is not.

Given a string representing the starting number and a maximum number of changes allowed, create the largest palindromic string of digits possible or the string -1 if it's impossible to create a palindrome under the contstraints.

**Function Description**

Complete the highestValuePalindrome function described below to determine the best palindrome, if possible.

|  |
| --- |
| highestValuePalindrome(string: s, integer: n, integer: k) |
| Parameters:  *s*: string representation of starting number     *n*: length of *s*     *k*: maximum number of changes*C*  Returns: string, highest value palindrome or "-1" if impossible |

**Constraints**

* Each character  in the number is an integer where .

**Raw Input Format**

The first line contains two space-separated integers,  (the number of digits in the number) and  (the maximum number of changes allowed), respectively.   
The second line contains an -digit string of numbers that you are to attempt to make palindromic.

**Sample Input 0**

4 1

3943

**Sample Output 0**

3993

**Explanation 0**

There are two ways to make  a palindrome by changing no more than  digits:

, so we print .

**Sample Input 1**

6 3

092282

**Sample Output 1**

992299

**Sample Input 2**

4 1

0011

**Sample Output 2**

-1

Solution:

#include <cmath>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

//return 1, if the string is palindrome

int isPalindrome(string str, int length)

{

int start=0, count=0, end=length-1;

while(start<=end)

{

if(str[start]!=str[end])

count+=1;

start+=1;

end-=1;

}

return count;

}

int main()

{

int length, permit, count=0, start, end;

string str, dummy;

cin>>length >>permit >>str;

//the tracking string, that tells which bits

//are changed after the palindrome operation

//for the purpose of maximizing the palindrome

dummy=str;

//Minimum Swap operations required to make

//the given string a Palindrome

count = isPalindrome(str, length);

//Maximizes if the string is already a palindrome

if(count == 0)

{

if(permit %2 != 0 && length%2 != 0)

{

str[length/2]='9';

permit-=1;

}

start=0; end=length-=1;

while(permit>1 && start <= end)

{

if(str[start]!='9')

{

str[start]=str[end]='9';

permit-=2;

}

start+=1;

end-=1;

}

cout<<str;

return 0;

}

//not enough operations, ABORT !

if(permit < count)

{

cout<<"-1";

return 0;

}

//Constructing palindrome

if(permit >= count)

{

//cout<<"Inside the constructing palindrome block"<<endl;

start=0;

end=length-1;

while(start <= end)

{

if(str[start] != str[end])

{

if(str[start]>=str[end])

str[end]=str[start];

else

str[start]=str[end];

}

start+=1;

end-=1;

}

}

//constructing the biggest palidrome

if(permit > count || count==0)

{

int extra = permit - count;

//Finding & swapping letters not equal to '9'

start=0;

end=length-1;

while(start <= end && extra>=1)

{

//letter found

if(str[start]!='9')

{

//this pair was changed, so onlu

//one operations is required to

//make the pair -->'9' & '9'

if(dummy[start] != dummy[end])

{

str[start]='9';

str[end]='9';

extra-=1;

}

//since min 2 operations are needed

else if(extra>=2)

{

str[start]='9';

str[end]='9';

extra-=2;

}

}

start+=1;

end-=1;

}

}

cout<<str;

return 0;

}

`’Anshul AgGarwal