C-Style String

→ A string is 1-D of char. array.

→ they are seq. of char. that is treated as single data item & terminated by null char '\0'.

- Char ch[5] = "Harsh"; → error (no space for null).
- Char ch[5] = "Haro"; → valid
- Char ch[5] = {'H', 'a', 'r', 's', 'h', '\0'}; → valid String
- Char ch[4] = {'H', 'a', 'r', 's', 'h'}; → think you are using this as array

Headerfile → <Cstring> for strcpy, strcmp etc.

String class in C++

String class store char. as a seq. of bytes, with functionality of allowing access to single byte character.

Char array	String
Size of char array is allocated strictly.	In case of String, memory allocate dynamically
Faster	slower
don't have much inbuilt func".	have lot of in-built function.

- To use String add → #include <string>
- then declare & initialize String variable

Type conversion

Example

```
#include<string>
using namespace std;
int main() {
    string str1 = "Harsh";
    string str2 = "course";
    string str; cin >> str;
    cout << " simple string = " << str;
    return 0;
}
```

Input
Data structure
output
Data

→ Cin considers every space as a terminator, ∴ we're getting only Data as output.

⊙ How to get full sentence ?

function in string class :-

1) getline() → stream of char as input (Data Structure)
2) push_back() → append char at the end of my string
3) pop_back() → Data struct → pop_back() → Data struct

Example

```
int main() {
string str2;
 getline (cin, str2);
cout << " str2 = " << str2 << endl;
 string str1 = "Harsh";
 str1.push_back("V");
 cout << "str1 = " << str1;
 str1.pop_back();
 cout << " str1 = " << str1;
```

I/P
Data structure
O/P
Data structure

HarshV

Harsh

functions in string

(1) capacity()

str.capacity() is diff. b/w then length()

capacity >= length, bcoz string is dynamic, if it has 10 in starting we can add more in it and increase its length.

(2) length() → show perfect length of string

(3) resize() → if it is 12 → resize to 14

Example

```
String str = "Harshvardhan";
cout<<" capacity is :"<< str.capacity();
cout<<" length is:" << str.length();
```

capacity is : 15
length is : 11

```
cout << " resized :" << str.resize(9) << str; → Harshvard.
```

Operator apply on string (str1 = 'A', str2 = 'B')

= assignment (str1 = str2)

+ → concatenation (s1 = 'A', s2 = 'B' → s1+s2 = 'AB')

+= → concatenation assignment (str1 += str1 + str2) (str1+=str2)

== → equality (campare alphabetically) (if (str1 == str2))

!= → inequality (if (str1 != str2))

< → less than

> → greater than

<= → less than or equal to

>= → greater " "

[] → subscript

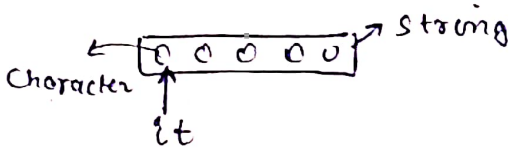
<< → output

>> → Input

Iterator function

→ an iterator is an object (like pointer) that points to an element inside the container.

→ We can use iterator, to move through the contents of container



→ while using iterator, no need to know size of container, end() func" take care of that.

→ It provide dynamically add or remove ele. from container as and when we want with ease.

functions of iterator

(1.) begin() (2) end() (3) rbegin() (4) rend()

String - (r)emov(e) → rend() return last element

rbegin() return first element

> If we want access or print string in reverse manner we use rbegin() & rend, else we use normal begin() & end()

Example :- (simple begin() & end())

```
int main() {
    string :: iterator sit;  Str1 = "Harshvp";
    cout<< " with begin() & end() : ";
    for (sit = Str1.begin(); sit != Str1.end(); sit++)
    {
        cout << *sit; }
    cout << endl;
    return 0;
}
```

with begin() & end() : Harshvp

Example (rbegin() and rend())

```
int main() {
string :: reverse_iterator rsit;
String str1 = "Harshup";
cout << "String with rbegin() & rend is :";
for (rsit = ~~rsit~~ str1.rbegin(); rsit != str1.end(); rsit++) {
        cout << *rsit; }
  cout << endl;
} return 0;
}
```

String with rbegin : pvhgraH

⊙ copy func" in string

Syntex : `copy("char array", length, position)`

```
String str = "function";
Char ch[5]
```

→ if length = 5 and pos^n = 2

Char arr → [nctio] ← from 2 to 5 length it'll print

```
Str.copy(ch, 5, 2);
cout << ch;
```

⊙ Swap()

str1 = 'A', str2 = 'B'.

```
Str2.swap(Str1);
```

Str1 = 'B', Str = 'A'.

⊙ find("string") function

Searches the string for the first occurrance of substring specified in arguments, it return 1st occurrance of sub.str.

⊙ rfind("string") → return Last occurrana of string

⊙ find_first_of("string") : Searchus for the string fo the first char. that match any of the char specify in its args.. (return pos" of 1st char that match)

eg. String str = Specified

Substring ⓒif → 3rd index

⊙ find_Last_of("string") it return pos" of last char that matches with substring in args.

eg: ⇒ String str = Spⓔcifiedⓔ

Substr = "ifiⓔ" → 2nd index

⊙ insirt (pos_to_begin, string_to_insirt)

⊙ clear()

⊙ empty() - return in true and false

Example: String Str1 = "Harsh";
0 1 2

⊙ Str1.insert(3, "vard") → Harvardsh

⊙ if(str1.empty()){} else { }.

⊙ Str1.clear(). → becomes empty

Ques → first non-repeating characters (Problem 1)

Given string S, print first non-repeating character of S.

$1 <= |s| <= 10^6$, all lowercase character

Example

String- "b c a d b c a e" → d is 1st non repeating

String: "harsh → a is 1st non-repeating

Brute Force :-

1. How many times a char is occouring ?
2. Yes, we can directly say which char occour 1s only.

b c a d b c a e : we check every element with all other
: whether its repeating or not

→ once you found char which can't be find anywhere, we stop and print that char.

→ If repeating then, move 1 step ahead.

Pseudo code

```
for(i=0; i<n-1; i++)                    (2)
    char ch= str[i]; flag = 0;
    for(j=i+1; j<n-1; j++)              (3)
        if(str[j] == ch)
            break; flag = 1; break;
                                        (3)
    if(flag == 1) cout<<" Found char Repeating char";
    else if(flag == 0)
            cout<<" got the non repeat "<< ch;
            break;
                                        (1)
```

TC - $O(n^2)$
SC - $O(1)$

here 0 to $< n-1$ => $(10^6 - 1) \cong 10^6$
$i+1$ to $10^6 - 1$ => $\cong 10^6$
=> $10^6 \times 10^6 = 10^{12}$

Optimized approch

b c d a b c e →

b - 2
c - 2
d - 1
a - 1
e - 1

by using ascii conversion
char to integer

0 ---- 127

Map a to z								
0	1	2	3	4	---	25		
↓	↓	↓	↓	↓		↓		
a	b	c	d	e	----	z		

→

a = 97 → a = 0 → 97 - 97
b = 98 → b = 1 → 98 - 97
c = 99 → c = 2 → 99 - 97
d = 100 → d = 3 → 100 - 97

Simply - (str[i] - 97)

Pseudo code

① take string as input ② declar frequency array freq[26]; = {0}

③ fill this array → for(i=0; i<str.length; i++) {

④ freq [str[i] - 97]++; }

b - 2
c - 2
d - 1
a - 1
e - 1

⑤ Run loop on string →

for (i=0; i<str.length(); i++) {

⑤

if (freq[str[i] - 97] == 1) {

return str[i];

break; }

}

→ Sum 2 no. representing string

Given 2 string, add 2 no. given as string.

Example

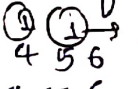
Str1 = "123", Str2 = "456"

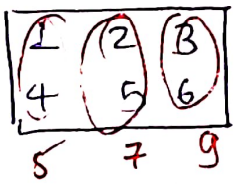
Str1 + Str2 = "579" → [(1+4).(2+5)(3+6)]

Logic building

Str = "123" → these all are character, & once we find out how can we add char, then we can solve

take care of carray value & quotient (single value)





we'll add each no. independently, after

ch → int → sum → change to char → store to str.

char "0", "1", ..., "9" → we'll use frequency array. using ASCII

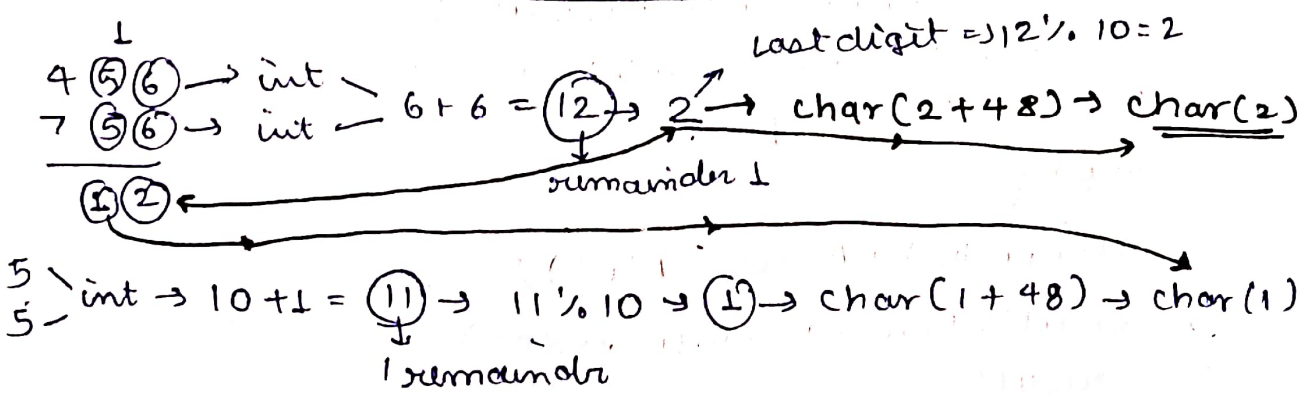
int → 0 1 ... 9

char ↔ int	
0 → 48	48 - 48 = 0
1 → 49	49 - 48 = 1
2 → 50	50 - 48 = 2
3 → 51	51 - 48 = 3
⋮	
9 → ..	

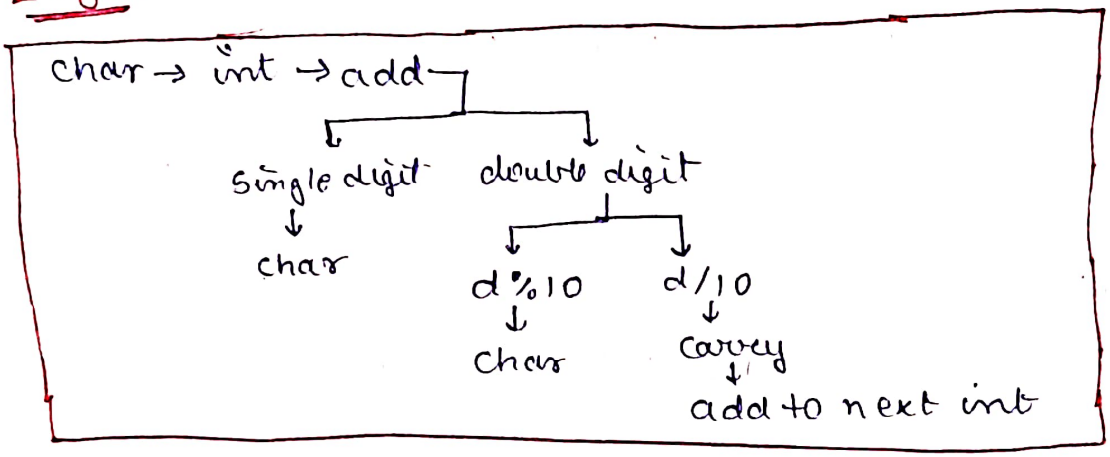
int to char → add 148 ≈ (char)(1 + 48)

(49)

explicit type cast



Logic



Pseudo code

```
int main() {
ctn string str1, str2;
    cin >> str1 > str2;

if (str1.length() > str2.length())
      swap (str1, str2);

int n1 = str1.length();
int n2 = str2.length();

reverse (str1.begin(), str1.end());
reverse (str2.begin(), str2.end());

int carry = 0;
string result;
for (int i=0; i<n1; i++) {
  int sum = ((str1[i]-'0')+(str2[i]-'0')+ carry);
  result.push_back(sum%10 + '0');
  carry = sum/10;
}
for (int i=0; i<n2; i++) {
int sum = ((str2[i]-'0') + carry);
  result.push_back(sum%10 + '0');
carry = sum/10;
}
if (carry) {
  result.push_back(carry+'0');
}
reverse (result.begin(), result.end());
cout << result;
return 0;
}
```

Ques → Aditya and Danish

Both play N games of chess in row, and note down winner as A or D, there is no tie. tell us who is great player.

(*) Example

N - length of string string - "AADADAA" output - Aditya

6 DADADA output - AdiDan

(*) Intution :-

Loop on string and count As and Ds and store if count.

if (Acnt > Dcnt) → Adity, else if (Dcnt > Acnt) → Danish

else → AdiDan

TC - O(n)

Ques → Vowel and consonents

Given a string, count no. of vowel and consonents presents in a strung

Example

String in uppercase :-

HARSHVARDHAN → o/p 3 (vowel) 8 (consonents)

(*) Intution

2 variable to stor count of vowel & consonent

VC and CC=0;

if (str[i] == 'A' || 'E' || 'I' || 'O' || 'U') VC++;

else CC++;

return count of both;

Ques → Vowels and consonants

Prefix and Suffix

Given a string, task is first print all prefix in each new line then print all suffix in new line.

⊛ Example

String = 'abc' →

a
a b
a b c
] Prefix

c
b c
a b c
] Suffix

⊛ Intution

for prefix print all sub-string of str in forward and for suffix print all substring in reverse order.

⊛ Code

```
String str;
cin>>str;
//print all substring in forward
for(int i=0; i<str.length(); i++)
{  for( int j=0;  j<=i;  j++)
   {
      cout<<str[j];
   }
 cout<<endl;
}
// print all sub string in reverse
for(int i=str.length()-1;  i>=0;  i--)
{
    cout<<&str[i]<<endl;
}

return 0;
}
```

this & responsible for printing suffix

...es → playing card

In this game, 52 card rangk from (2,3,4,5,6,7,8,9,10=T, J, Q, K, A) and Diamond - D, club - C, spade - S, Heart - H .. 4 suit.
every player has 5 card in hand rest with 1 open in front, a player can only play, if open card has same rank or of same suit. now its your friend's chance find out if he could play atleast 1 card?

(*) Example

I/P: AS → card on table
2H 4C TH JH AD → card in hand
O/P: Yes

(*) Intution

1. if we find A or S in string we can play
2. every input of string has 2 paramit (Rank) (suit) same with card on desk
3. Strati for 5 times only bcoz, 5 card in hand.
4. chick if table_card[0] == card_hand[0] || [1] == [1].

(*) Pseudo Code

```
int c=0;  string s; string s1;
cin >> s; // card ~~on table~~ hand
for( i=0 → i<5) {
    cin >> s1; // card on table
    if( ( s[0] == s1[0] ) || ( s[1] == s1[1] ) )
        c++;
}
if (c>0)  cout << "YES";
else cout << "NO";
```

Ques → Nobel Vowel

Nobel Vowel - There has a to a vowel after every consonant but there can be any letter after vowel.

condition : there can be any letter after consonant 'n' only. given a string check for nobel vowel.

Example

String = aeiou , String = "cefduo"

O/P = Yes O/P = NO

Intution

Make a isVowel() to check for vowel for every char of string.

take initially check=0; & while (i != str.length()) {

if (check==0) → if (isalpha(str[i])) → {if (isVowel(str[i]))

check=0;

else {

if (data[i]=='n')

check=0;

else

} check=1;

}

else check=-1;

else we again check for (check==1) : {

if (isVowel(str[i])) check=0;

else

check=-1; }

else if

check=-1;

break;

i++;

}

if (check==0) cout<<"YES";

else

cout<<"NO";

TC - O(n)

SC - O(1)

Q5 → Decreasing Order String

Sort the given string, in lexicographical-decreasing order.

(*) Example

S = "Harshvardhan" → aaadhhhnnrrsv

(*) Intution

(1) take alphabet and its count, in a freq. array = {0}.

(2) for loop' for entering all element of string in freq[ ].
as str[i] - 'a' bcoz if we need 0 'a' - 'a',
1 → 'b' - 'a', 2 → 'c' - 'a' ---.

(3) nested for loop, in reverse of ~~take from~~ ~~1 for~~ 1 for iterating back to from on MAX_char, which is = 26 (all 26 alphabet.).

(4) another loop of size of string, & and we explicitly convert it as (char)('a' + i);

(*) Snippet

TC - O(n²)

```
const int MAX_CHAR = 26;
void sorting(string str) {
int freq[MAX_CHAR] = {0};
for(int i=0; i<str.length(); i++)
    freq[str[i] - 'a']++;
for (i = MAX_CHAR-1; i >= 0; i--) {
    for (int j = 0; j < freq[i]; j++) {
        cout << (char)('a' + i);
```

explicit typecasting

Ques → Duplicate Character

Given string, print characters whose freq. count is more than once. and if not such case then return -1.

Example

	output
Harshard →	r=2 a=2 r=2
algorithm →	-1

(*) Intution

1. create map with char and its freq. count.
2. using an iterator, iterate on map and, if (count > 2) print.
3. if not then -1.

(*) Implementation

```
string s;
cin >>s;   int flag = 0;

map<char, int> m;
map<char, int>::iterator it;
for(int i=0; i<s.length(); i++) {
    m[s[i]]++; }
if(m.size()==0) cout<<"-1"<<endl;
else {
    for(it=m.begin(); it!=m.end(); it++)
        if(it->second>1) {
            flag = 1;
            cout<<it->first<<"="<<it->second<<" "; }
if(f==0) { cout<<"-1"; }
    cout<<endl;
}
```

TC - O(n)
SC - O(n)

Files → Aman String

find no. of such pair of indices (i,j) - $(1 \le i \le j \le |s+1|)$. such that string $s(i,j)$ contains at least once string "aman" as substring.

Example

String = amanmgnaa ans = 6

pairs are: (1,4), (1,5), (1,6) (1,7) (1,8) (1,9)

Implementation

```
int main() {
 string s;
 vector<int> v;
 cin >> s;
long long n=s.size(), x=1, y, ans=0;
for(int i=0; i+3<n; i++) {
   if(s[i]=='a' && s[i+1]=='m' && s[i+2]=='a' && s[i+3]=='n')
   v.push_back(i+1);
}
for(int i=0; i<v.size(); i++) {
    y = n - v[i] - 2;
    if(i)
      {x = v[i-1]+1; }
    x = v[i] - x + 1;
    ans += (x*y);
}
 cout<<*ans<<endl;
 return 0;
}
```

Ques → equal String

n strings of equal size is given, you can perform the operation as remove 1st element and append to end of string. Return min no. of operation needed to make string equal.

Example

Str1 : abcde
Str2 : cdeab

→ abcde → (1) bcdea → (2) cdeab = Str2

Ans = 2

Intution-1

We can solve this, by using LCS of 2 string

example - abcd cdeab LCS = abd => Ans = 2

2C1 2C2

(counter update when we delete so, 2C1 + 2C2)

Approach-2 :-

```
int minOperation(string arr[], int n) {
int ans = INT_MAX;
for(int i=0; i<n; i++) {  int curr_count = 0;
/* consider arr[i] as target string, & count rotat" req. to make
 all other str. same as Str[i] */
 for(int j=0; j<n; j++) {  String temp= arr[j] + arr [j];
/* find func" return index where we found arr[i], which is
actually count of move-to-front operation */
int index = temp.find(arr[i]);
/* if 2 str are not rotation of each other, can't make them same. */
  if (index == String::npos)  return -1;
  curr_count += index;
  ans = min(curr_count, ans);
  }
  return ans;
}
```

15 - Chatur's Exam paper

Uncle changed all words of Chatur's all word in palindrome. Follow 2 rules :-

(1) In 1 operation, he can only reduce value of alpha by 1, Eg. he can change d to c, but can't d to b or d c to d

(2) Alpha. 'a' can't reduce, further.

each reduct^n count as single operation. min. no. of opert^n required to convert given string in palindrome.

* Example

Str1 : abc → Ans 2 ∵ c ①→ b ②→ a ∴ 2 operation

Str2 : abcba → Ans 0

Str3 : a b c d → Ans 4 ∵ d ①→ c ②→ b ③→ a, Ans => 4
c ④→ b

* Intution :-

We know that to make c to a, we have to subtract as a = 97, c = 99 ⇒ 99 - 97 = 2 or 99-2=97
∴ we need absolute diff. b/w 1st & Last char and ↑se ith index & ↓se jth index (from last)

* Code

```
int main() {
    string s; cin >> s;
    int i = 0; j = s.length() - 1; int ans = 0;
    while (i < j) {
        ans += abs(s[i] - s[j]);
        ++i; --j;
    }
    cout << ans << endl;
}
```

Ques → Fake Password

Check whether original password can be obtain by rotating fake password by 2 places.

⊛ Example

Str1: string → string → gstrin → ngstrin => true (Yes)
Str2: ngstri

⊛ Intution (using queue) (right rotation)

1) If size of both string in unequal, then can't possible
2) push str1 in queue 1, & string 2 in queue 2.
3) while, till no. of operation less than size of string
   eg. if k = 6 (size of string), then while (k--).
4) pop front ele. of queue and push it back, & check if ($q_1 == q_2$).

TC - O(n1+n2)
SC - O(n)

⊛ Implementation

```
bool checkRotation(string s1, string s2) {
    if (s1.size() != s2.size()) return false;
    queue<char> q1; for(auto i : s1) q1.push(s1[i]);   // s1.size()
    queue<char> q2; for(auto i : s2) q2.push(s2[i]);   // s2.size()
    int k = str2.size();
    while(k--) {
        char ch = q2.front();
        q2.pop();
        q2.push(ch);
    }
    if (q1 == q2) return true;
    return false;
}
```

Rotation of string from right

```
int main() {
    string str1 = "string";
    string str2 = "gstrin";
    if (checkR(str1, str2))
        cout << "Yes" << endl;
    else
        cout << "NO" << endl;
}
```

Ques → Valid String

Fix the bracket sequence by only moving one of the bracket in the sequence. on the following condit".

(1) bracket sequence (S) is empty.

(2) S is equal to "(t)", where 't' is valid seq.

(3) S is equal to "($t_1 t_2$)", where t_1 & t_2 are valid concatination sequence.

Check if seq. can be made, just by moving 1 bracket.

⊛ Example

Str1: (()) → Yes

Str2:))((→ NO

⊛ Implementation

"Our func" will return true, if seq. can be balanced by change post "at most brac"

```
bool balance( string s, int n) {
  /* Odd length str can never be balanced */
  if (n%2==1) return false;
/* add '(' in the beg. & ')' in the end */
  string k = "("; k += s + ")";
  vector<string> v;  int cnt = 0;
  for (int i=0; i<k.length(); i++) {
/* If its an open bracket, append temp string 'v' */
  if (k[i]=='(') d.push_back("(");
/* if its closing bracket */
  else { if (d.size()!=0)
             d.pop_back();
/* there is an opening bracket to match it with */
/* if no open bracket to match it with */
  else {
      return false;
  } }
  if (d.empty) return true;
  return false; }
```

Ques → Binary Addition

2 string A & B given in binary form, return its sum in binary string.

✱ Example

A: 1011
B: 1010

⇒

```
  1011
+ 1010
------
 10101
```

Ans ⇒ 10101

✱ Implementation

```
String addBinary (string A, string B) {
  string res = "";
  int i = A.length()-1; int j = B.length()-1; int carry = 0;
  while ( i>=0 || j>=0 || carry>0) {
    int ival = i>=0 ? A.at(i) - '0' : 0;
    int jval = j>=0 ? B.at(j) - '0' : 0;
    i--; j--;
  }
  int sum = ival + jval + carry;
  res += char (sum%2 + '0'); carry = sum/2;
  reverse (res.begin(), res.end());
  int zero = 0; int k = 0;
  while (res.size()>1) {
    if (res[k] == '0')
      res.erase (res.begin()+k);
    else
      break;
  }
  return res;
}
```

Ques → Longest Unique

Given string, find longest substring with unique char.

⊛ Example

Str: prepration → epration → 8 Ans

Str: Amazon → 5 & ans (all are unique)

⊛ Brute force

Find all string without duplicate, & return max-length substring. TC - $O(n^3)$

⊛ Optimal :- $O(n^2)$

```
int longest (string str) {
    int n = str.size(); int res = 0;
    for (int i=0; i<n; i++) {
        /* default values in visited vector 'v' is false */
        vector<bool> v(256);
        for (int j=i; j<n; j++) {
            /* if curr. char visited break the loop */
            if (v[str[j]] == true) break;
            /* else update result if this window is large, & mark curr char visited */
            else {
                res = max(res, j-i+1);
                v[str[j]] = true;
            }
        }
        /* Remove 1st char of previous window */
        v[str[i]] = false;
    }
    return res;
}
```

Ques → Minus - Minus is plus (Hard)

(-) * (-) makes plus, given 2 string s1 & s2 both contain '-' & '+' symbol only, can we obtain s2 by doing operation on s1 by 0 or more times?

Operation

choose any 2 adj. (-) sign & replace with (+) sign, 1 such operation reduce length of string by 1.

Ques → Multiply two number (Hard)

2 string s1 & s2 given, print prod of s1 & s2, but output is in string only. (in integer.)

example: s1: 3
s2: 2 ⇒ s1*s2 = 6 (string)
(3)×(2)

Implementation:

```
String multiply(string n1, string n2) {
if(n1=="0" || n2=="0") return 0;
int l1 = n1.size()-1; int l2 = size()-1; int carry=0;
String ans, prev="0"; int i=0;
while(i1>=0) {  char ch= n1[i]; int carry=0;
 int tn2 = l2;  string temp="";
 while(tn2>=0 || carry!=0) {
 char b; if(tn2<0) b='0'; else b= n2[tn2];
 temp+= ((b-'0') * (n1[l1]-'0') + carry) % 10 + '0';
 carry = ((b-'0') * (n1[l1]-'0') + carry) /10;
 tn2--;
}
reverse(temp.begin(), temp.end());
return ans;
}
```

3