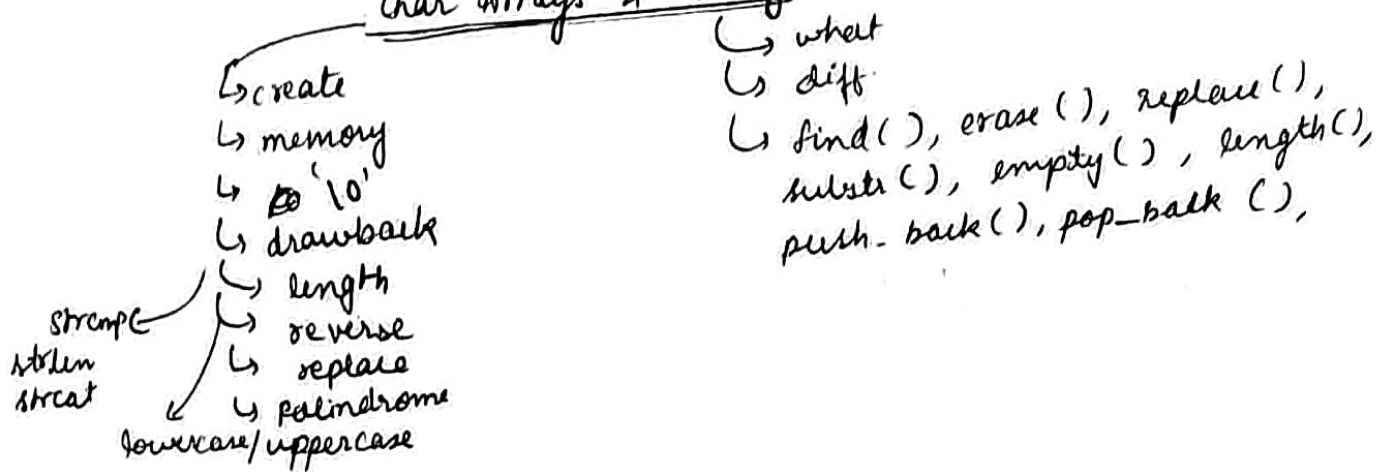


Char Arrays & strings



Questions-

- Valid Palindrome
 - Remove all adjacent duplicate
 - Remove all occurrence of substring.
 - Min. time Difference
 - Count palindromic substring.
- Brute force $\rightarrow O(n^3)$
 2-pointers
 odd substr $\rightarrow O(n^2)$
 even substr

① Sort function and custom comparator \rightarrow

```

string s = "babbar"
sort(s.begin(), s.end());
cout << s << endl;
  
```

// o/p \rightarrow aabbbr
 \downarrow
 lexicographical order.

This string is converted into ascending order, but we want it in descending order. How to do this?

\rightarrow by custom comparator.

```

bool cmp(char first, char second){
    return first > second;
}
  
```

```

int main(){
    string s = "babbar";
    sort(s.begin(), s.end(), cmp);
    cout << s;
}
  
```

// o/p \rightarrow rbbbaa

B	A	B	B	A	R
0	1	2	3	4	5

→ sort (s.begin(), s.end())
 ↓ ascending order

A	A	B	B	B	R
0	1	2	3	4	5

→ custom comparator →
 bool cmp (char x, char y) {
 return ^{asc order} x < y;
 ^{disc order} x > y;
 }
 ↪ we don't need to pass send these chars.

Sorting → If we have two elements then sorting is to decide which element to keep before by comparing two elements.

vector<int> v {5, 3, 1, 2, 4};

sort (v.begin(), v.end());

for (auto i : v)
 cout << i << " ";
 } cout << endl;

// o/p → 1 2 3 4 5
 sorted in ascending order.

bool compare (int a, int b) {

 return a < b; → ascending order (By default)
 }

bool compare2 (int a, int b) {

 return a > b; → descending order.

{
 int main() {
 vector<int> v {5, 3, 1, 2, 4};

 sort (v.begin(), v.end(), compare);

 sort (v.begin(), v.end(), compare2);

// 1 2 3 4 5

// 5 4 3 2 1

HW
Ques - Help Ramu coding question.

Suggestion →

Whenever you have doubt about working of a function, just try ~~not~~ printing each step of that func.

eg → bool compare (int a, int b) {

 cout << "first no. is:" << a << endl;

 cout << "second no. is:" << b << endl;

 return a > b;

{
 main() {

 sort (v.begin(), v.end(), compare);

Usage →

⇒

2 Character sum → we have to sort a string according to their character sum.

"lowe"	"babbar"	"rahul"	"sunday"	"good"
--------	----------	---------	----------	--------

$$\text{lowe} = 12 + 15 + 22 + 5 = 54$$

$$\text{babbar} = 2 + 1 + 2 + 2 + 1 + 18 = 26$$

$$\text{rahul} = 18 + 1 + 8 + 21 + 1 = 49$$

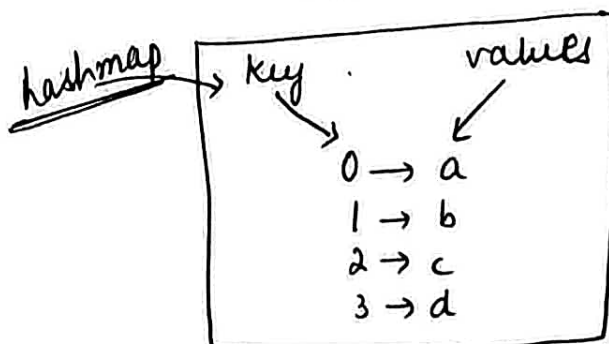
$$\text{sunday} = 19 + 21 + 14 + 4 + 1 + 25 = 79$$

$$\text{good} = 7 + 15 + 15 + 4 = 26 + 15 = 41$$

ans →

"babbar"	"good"	"rahul"	"lowe"	"sunday"
----------	--------	---------	--------	----------

① HashMap → A type of data structure in which ~~values~~ elements are stored in form of key-value pairs.



here key is numeric.
and value is character
type.

// creation of maps →
map <int, char> myMap;

myMap[0] = 'a';

myMap[1] = 'b';

myMap[2] = 'c';

cout << ~~myMap~~ "value : " << myMap[0] << endl; // 0/p → value: a

cout << "value : " << myMap[2] << endl;

// 0/p → value: empty.

Why we use map?

Because in unordered-map insertion and accessing time complexity is $O(1)$.

But here the T.C is $O(\log n)$.

auto keyword →

```
for (auto val : arr) {  
    cout << val;  
}
```

By using auto we do not need to tell explicitly the datatype of element.

① maximum find

int maxi = INT_MIN;

minimum find

int mini = INT_MAX;

→ Recommend to initialize with these.

INT_MIN and INT_MAX are macros.