1. Software Development
2. Back end Developer
3. Front End Developer
4. System Engineer
5. Software Engineer
6. Web developer
7. Software Development Engineer
8. Software Engineer Backend
9. Full Stack Developer

**Strings:-**

Strings

1. Reverse String

\*class Solution

{

public void reverseString(Char[] s)

{

int left=0, right=s.length-1;

while(left<right)

{

Char temp=s[left];

s[left++]=s[right];

s[right--]=temp;

}

}

}

\*class solution{

public void helper(char[] s, int left,int right)

{

if(left>=right)return;

char temp=s[left];

s[left++]=s[right];

s[right--]=temp;

helper(s,left,right);

}

public void reverseString(char[] s)

{

helper(s,0,s.length-1);

}

}

2.Reverse Integer

Class Solution

{

Public int reverse(int x)

{

Int rev=0;

While(int x!=0)

{

Int pop=x%10;

X=x/10;

If(rev>Integer.MAX\_VALUE/10 ||(rev==Integer.MAX\_VALUE/10 &&POP>7))

return 0;

If(rev<Integer.MAX\_VALUE/10 ||(rev==Integer.MAX\_VALUE/10 &&POP<-8))

return 0;

rev=rev\*10 +pop;

}

return rev;

}

}

3.First Unique Character in a String

Class Solution

{

Public int FirstUnique(String s)

{

HashMap<Character, Integer> count = new HashMap<Character,Integer>();

Int n=s.length();

for(int i=0;i<n;i++)

{

Char c=s.charAt(i);

Count.Put(c,count.getorDefault(c,0)+1);

}

for(int i=0;i<n;i++)

{

If(count.get(s.charAt(i)==1)

return i;

}

return -1;

}

}

4.Valid Anagram

Class solution

{

Public Boolean isAngaram(String s, String t)

{

If(s.length()!=t.length())

{

Return false;

}

Int[] counter=new int[26];

For(int i=0;i<s.length();i++)

{

Counter[s.charAt(i)- a ]++;

Counter[t.charAt(i)- a ]--;

}

For(int count:counter)

{

If(count!=0)

{

Return false;

}

}

return true;

}

}

5.Valid palindrome;

Class solution

{

Public boolean isPalindrome(String s)

{

For(int i=0, j=s.length()-1;i<j; i++,j--)

{

While(i<j && !character.isLetterOrDigit(s.charAt(i)))

{

i++;

}

While(i<j && !character.isLetterOrDigit(s.charAt(j)))

{

j--;

}

If(Character.toLowereCase(s.charAt(i))!=Character.toLowerCase(s.charAt(j)))

Return false;

}

return true;

}

}

6.Find the Index of the First Occurrence in a String

class solution

{

public int strStr(String needle,String haystack)

{

int m=needle.length();

int n=haystack.length();

for(int windowstart=0; windowstart<n-m;windowStart++)

{

for(int i=0;i<n;i++)

{

if(needle.charAt(i)!=haystack.charAt(windowstart +i))

{

break;

}

if(i==m-1)

{

return windowstart;

}

}

}

return -1;

}

}

7.Longest Common Prefix

class solution

{

public int longestCommonPrefix(String[] strs)

{

if(strs.length==0) return"";

String prefix=strs[0];

for(i=1;i<strs.length;i++)

while(strs[i].indexOf(prefix)!=0)

{

prefix=prefix.substring(0,prefix.length()-1);

if(prefix.isEmpty()) return "";

}

return prefix;

}

}

8.String to Integer

class solution

{

public int myAtoi(String input)

{

int sign=1;

int result=0;

int index=0;

int n=input.length();

while(index<n && input.charAt(index)=='')

{

index++;

}

if(index<n && input.charAt(index)=='+')

{

sign=1;

index++;

}

else if(index<n && input.charAt(index)=='-')

{

sign=-1;

index++;

}

while(index<n && Character.isDigit(input.charAt(index)))

{

int digit=input.charAt(index)-'0';

if ((result > Integer.MAX\_VALUE / 10) ||

(result == Integer.MAX\_VALUE / 10 && digit > Integer.MAX\_VALUE % 10)) {

return sign == 1 ? Integer.MAX\_VALUE : Integer.MIN\_VALUE;

}

result = 10 \* result + digit;

index++;

}

return sign \* result;

}

}

Reverse String:-

class solution

{

public void reverseString(char[] s)

{

int left=0, right=s.length-1;

while(left<right)

{

char temp = s[left];

s[left]=s[right];

left++;

s[right]=temp;

right--

}

}

First Unique Character in a String

class solution

{

public int firstUniqueCharcter(string s)

{

HashMap<charcter,Integer> count=new HashMap<character,Integer>();

int n=s.length();

for(int i=0;i<n;i++)

{

char c=s.charAt(i);

count.put(c,count.get(c,0)+1);

}

for(int i=0;i<n;i++)

{

if (count.get(s.charAt(i))==1)

return i;

}

return -i;

}

}

**Arrays:-**

**Remove Duplicates from Sorted Array**

Class solution

{

Public int removeDuplicate(int[] nums)

{

int k=1;

for(int i=1;i<nums.length;i++)

{

if(nums[i-1]!=nums[i])  
{

nums[k]=nums[i];

k++;

}

}

return k;

}

}