DAY 8

INTERVIEW BIT PROBLEMS :

1. Minimum Characters required to make a String Palindromic

Given an string A. The only operation allowed is to insert characters in the beginning of the string.
Find how many minimum characters are needed to be inserted to make the string a palindrome string.

Input Format
The only argument given is string A.
Output Format
Return the minimum characters that are needed to be inserted to make the string a palindrome string.

For Example
Input 1:
A = "ABC"
Output 1:
2
Explanation 1:
Insert 'B' at beginning, string becomes: "BABC".
Insert 'C' at beginning, string becomes: "CBABC".

Input 2:
A = "AACECAAAA"
Output 2:
2
Explanation 2:
Insert 'A' at beginning, string becomes: "AAACECAAAA".
Insert 'A' at beginning, string becomes: "AAAACECAAAA".

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @return an integer
    def solve(self, A):
        rev=A[::-1]
        if rev==A:
            return 0
        for i in range(1,len(rev)):
            if rev[:i]+A==rev+A[-i:]:
                break
        return i
```

C++

```
int Solution::solve(string A) {
    int s=0,e=A.length()-1;
    int temp=e;
    while(s<=e){
        if(A[s]==A[temp]){
            s+=1;
            temp-=1;
        }
        else{
            s=0;
            temp=--e;
        }
    }
    return A.length()-(e+1);
}
```

2. Implement StrStr

strstr - locate a substring (needle) in a string (haystack).

Returns the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

CODE :

PYTHON 1

```
class Solution:
    # @param A : string
    # @param B : string
    # @return an integer
    def strStr(self, A, B):
        if not A or not B:
            return -1
        else:
            #splits the A into two strings like length of string before B and after B
            x = A.split(B)
            if len(x[0]) == len(A):
                return -1
            else:
                return len(x[0])
```

PYTHON 2

```
class Solution:
    # @param A : string
    # @param B : string
    # @return an integer
    def strStr(self, A, B):
        return A.find(B)
```

C++

```
int Solution::strStr(const string A, const string B) {
```

```
        int begin=0;
        int temp_s=begin;
        int j=0;
        while(temp_s<A.size()&&j<B.size()){
            if(A[temp_s]==B[j]){
                j+=1;
                temp_s+=1;
            }
            else{
                j=0;
                temp_s=++begin;
            }
            if(j==B.size()){
                return begin;
            }
        }
        return -1;
}
```

3. Compare Version Numbers

Compare two version numbers version1 and version2.
If version1 > version2 return 1,
If version1 < version2 return -1,
otherwise return 0.
You may assume that the version strings are non-empty and contain only digits and the . character.
The . character does not represent a decimal point and is used to separate number sequences.
For instance, 2.5 is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.
Here is an example of version numbers ordering:
0.1 < 1.1 < 1.2 < 1.13 < 1.13.4

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @param B : string
    # @return an integer
    def compareVersion(self, A, B):
        new_A=A.split('.')
        new_B=B.split('.')
        new_A=[int(a) for a in new_A]
        new_B=[int(b) for b in new_B]
        a=len(new_A)
        b=len(new_B)
        if a>b:
            for i in range(b,a):
                new_B.append(0)
        elif b>a:
            for i in range(a,b):
                new_A.append(0)
        for i in range(len(new_A)):
            if new_A[i]>new_B[i]:
                return 1
            elif new_A[i]<new_B[i]:
                return -1
        return 0
```

C++

```
int Solution::compareVersion(string A, string B) {
    unsigned long long int v1=0,v2=0;
    int i,j;
    int a=A.length();
    int b=B.length();
    for(i=0,j=0;(i<a || j<b);){
```

```
            while(i<a && A[i]!='.'){
                v1=v1*10+(A[i]-'0');
                i+=1;
            }
            //cout<<v1;
            while(j<b && B[j]!='.'){
                v2=v2*10+(B[j]-'0');
                j+=1;
            }
            //cout<<v2;
            if(v1>v2){
                return 1;
            }
            if(v2>v1){
                return -1;
            }
            v1=0;
            v2=0;
            i+=1;
            j+=1;
        }
        return 0;
}
```

4. Atoi

Implement atoi to convert a string to an integer.
Example :
Input : "9 2704"
Output : 9

Questions:
Q1. **Does string contain whitespace characters before the number?**
A. Yes

Q2. **Can the string have garbage characters after the number?**
A. Yes. Ignore it.

Q3. **If no numeric character is found before encountering garbage characters, what should I do?**
A. Return 0.

Q4. **What if the integer overflows?**
A. Return INT_MAX if the number is positive, INT_MIN otherwise.

CODE :

PYTHON

```
class Solution:
    # @param A : string
    # @return an integer
    def atoi(self, A):
        n=len(A)
        if n==0:
            return 0
        out=0
        sign=1
        i=0
        if A[0]=='-':
            sign=-1
            i+=1
        if A[0]=='+':
            i+=1
        for j in range(i,n):
            if not(A[j]>='0' and A[j]<='9'):
                if out==0:
                    return 0
```

```
            else:
                if (-1*pow(2,31)<sign*out<pow(2,31)-1):
                    return sign*out
                else:
                    if sign==-1:
                        return -1*pow(2,31)
                    else:
                        return pow(2,31)-1
        out=out*10+(ord(A[j])-ord('0'))

if (-1*pow(2,31)<sign*out<pow(2,31)-1):
    return sign*out
else:
    if sign==-1:
        return -1*pow(2,31)
    else:
        return pow(2,31)-1
```