Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe - Strings

1. Reverse words in a given string

```
Input: s = “geeks quiz practice code”
Output: s = “code practice quiz geeks”
```

In [ ]:
```
void reverse(string str){
    int start = 0;

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

        if(str[i] == ' '){
            reverse(str.begin()+start,str.begin()+i);
            start = i;
        }

    }

    reverse(str.begin()+start,str.end());

    reverse(str.begin(),str.end());
}
```

2. Permutations of a string

Write a program to print all permutations of a given string

In [ ]:
```
void permutations(string s){

    dfs(s,0);

}

void dfs(string s, int pos){

    if(pos == s.size()-1)
        cout<<s;
    else{
        for(int i=pos; i < s.size(); i++){
            swap(s[i],s[pos]);
            dfs(s,pos+1);
            swap(s[i],s[pos]);
        }
    }

}
```

3. Longest Palindrome in a string

In [ ]:
```
string findPalindrome(string s){
    int start = 0, maxlen = 0;

    for(int i=0;i<s.size();i++){

        int len1 = expandAroundCenter(s,i,i);
        int len2 = expandAroundCenter(s,i,i+1);

        int len = max(len1,len2);

        if(len > maxLen){
            len = maxLen;
            start = i - (len-1)/2;
        }

    }

    return s.substr(start,maxlen);
}

int expandAroundCenter(string s, int i, int j){

    while(i>=0 && j<s.length() && s[i]==s[j])
        i--,j++;

    return j-i+1;
}
```

3. Check if string is rotated by two places

In [ ]:
```
bool isRotated(string str1, string str2)
{
    if (str1.length() != str2.length())
        return false;

    string clock_rot = "";
    string anticlock_rot = "";
    int len = str2.length();

    // Initialize string as anti-clockwise rotation
    anticlock_rot = anticlock_rot +
                    str2.substr(len-2, 2) +
                    str2.substr(0, len-2) ;

    // Initialize string as clock wise rotation
    clock_rot = clock_rot +
                str2.substr(2) +
                str2.substr(0, 2) ;

    // check if any of them is equal to string1
    return (str1.compare(clock_rot) == 0 ||
            str1.compare(anticlock_rot) == 0);
}
```

4. Roman to Integer

In [ ]:
```
int romanToInt(string s) {
    map<char,int> mapping ={{'I',1},{'V',5},{'X',10},{'L',50},{'C',100},{'D',500},{'M',1000}};
    int i = s.length()-1;
    int num = 0;
    while(i>=0)
    {
        if(i<s.length()-1 and mapping[s[i]] < mapping[s[i+1]])
            num -= mapping[s[i]];
        else
            num += mapping[s[i]];
        i -= 1;
    }
    return num;
}
```

5. Anagram

In [ ]:
```
bool anagram(string a, string b){
    sort(a.begin(),a.end());
    sort(b.begin(),b.end());

    return a==b;
}
```

6. Remove Duplicates

In [ ]:
```
string removeDup(string s){

    int writeHead = -1, readHead = 0;

    while(readHead < s.length()){

        if(writeHead == -1 || s[readHead] != s[writeHead])
            s[readHead++] = s[++writeHead];
        else{

            while(readHead<s.length() && s[readHead]==s[writeHead]) readHead++;

            writeHead--;
        }

    }

    if(writeHead < 0)
        return -1;
    return s.substr(0,writeHead+1);

}
```

7. Minimum number of insertions reqd to make string palindrome

In [ ]:
```
int minInsertions(string s){

    int dp[s.size()][s.size()]={{0}};

    for(int len=1;len<s.size();len++)
    for(int i=0;j=len;j<s.size();j++)
        dp[i][j] = (s[i]==s[j])?dp[i+1][j-1]:min(dp[i+1][j],dp[i][j-1]);

    return dp[0][s.size()-1];
}

//Could also find Longest Common Subsequence of str and reverse of str
```

8. Longest Distinct Characters in the string

In [ ]:
```
int longestDistinctChar(string str){

    vector<int> lastIndex(26,-1);
    int maxLen = 0, start = 0;

    for(int end=0; i < str.size(); i++){

        start = max(start, lastIndex[str[end]-'a']);

        maxLen = max(maxLen, end-start+1);

        lastIndex[str[end]-'a'] = end;
    }

    return maxLen;
}
```

9. Implement atoi()

9. Implement atoi()

In [ ]:

```
int myAtoi(string str) {
    int num = 0, sign = 1, i=0, n = str.size();

    for( ;i<n && str[i]==' ';i++);

    if(i<n)
        if(str[i]=='-')
            sign = -1, i++;
        else if(str[i]=='+')
            i++;

    for(;i<n;i++)
        if(str[i]>='0' && str[i]<='9')
            if(num <= (INT_MAX-str[i]+'0')/10)
                num = num*10 + (str[i]-'0');
            else
                return sign==-1?INT_MIN:INT_MAX;
        else
            break;

    return num*sign;
}
```

10. Implement strStr()

In [ ]:

```

vector<int> generateLPS(string s){
    vector<int> lps(s.length(),0);
    int j=0;
    for(int i=1; i<s.length();)
        if(s[i]==s[j])
            lps[i++] = ++j;
        else if(j==0)
            i++;
        else
            j = lps[j-1];

    return lps;

}

int strStr(string haystack, string needle) {

    if(needle.size() > haystack.size())
        return -1;

    if(needle.size()==0) return 0;

    vector<int> lps = generateLPS(needle);

    int j=0,i=0;

    while(i < haystack.size())
        if(haystack[i] == needle[j]){
            i++,j++;
            if(j==needle.size())
                return i-j;
        }
        else if(j==0)
            i++;
        else
            j = lps[j-1];

    return -1;

}
```

11. Longest Common Prefix

In [ ]:
```
string longestCommonPrefix(vector<string> words){
    string prefix = "";
    for(int i=0; words.size() > 0; prefix +=words[0][i++] ){
        for(int j=0; j < words.size(); j++)
            if(i >= word.size() || (j>0 && words[j-1][i] != words[j][i]))
                return prefix;
    }

    return prefix;
}
```