Longest Common Subsequence:

```
#include <bits/stdc++.h>

int lcs(int i,int j,string str1,string str2){
    if(i<0||j<0){
        return 0;
    }
    //match
    if(str1[i]==str2[j]){
        return 1+lcs(i-1,j-1,str1,str2);
    }
    else{
        return max(lcs(i-1,j,str1,str2),lcs(i,j-1,str1,str2));
    }

    //notmatch
}
int getLengthOfLCS(string & str1, string & str2) {
  int i=str1.size()-1;
  int j=str2.size()-1;
  return lcs(i,j,str1,str2);
}
```

Longest Palindromic Subsequnce:

```
#include <bits/stdc++.h>

int lcs(int i,int j,string str1,string str2){
    if(i<0||j<0){
        return 0;
    }
    //match
    if(str1[i]==str2[j]){
        return 1+lcs(i-1,j-1,str1,str2);
    }
    else{
        return max(lcs(i-1,j,str1,str2),lcs(i,j-1,str1,str2));
    }

    //notmatch
}
int longestPalindromeSubsequence(string s)
```

```
{
   string s1=s;
   reverse(s.begin(),s.end());
   string s2=s;
   int i=s1.size()-1;
  int j=s2.size()-1;
  return lcs(i,j,s1,s2);
}
```

Minimum insertions to make a string palindrome:

```
int lcs(int i,int j,string str1,string str2){
    if(i<0||j<0){
        return 0;
    }
    //match
    if(str1[i]==str2[j]){
        return 1+lcs(i-1,j-1,str1,str2);
    }
    else{
        return max(lcs(i-1,j,str1,str2),lcs(i,j-1,str1,str2));
    }

    //notmatch
}
int minimumInsertions(string &s)
{
    string s1=s;
   reverse(s.begin(),s.end());
   string s2=s;
   int i=s1.size()-1;
  int j=s2.size()-1;
  return s.size()-lcs(i,j,s1,s2);
}
```

Minimum Insertions and Deletions to Convert string A to string B:

```
int lcs(int i,int j,string str1,string str2){
    if(i<0||j<0){
        return 0;
    }
    //match
    if(str1[i]==str2[j]){
```

```
        return 1+lcs(i-1,j-1,str1,str2);
    }
    else{
        return max(lcs(i-1,j,str1,str2),lcs(i,j-1,str1,str2));
    }

    //notmatch
}
int canYouMake(string &s1, string &s2){
    int i=s1.size()-1;
    int j=s2.size()-1;
    return (s1.size()-lcs(i,j,s1,s2))+(s2.size()-lcs(i,j,s1,s2));
}
```

Shortest Common Supersequence:

```
class Solution
{
    public:
    //Function to find length of shortest common supersequence of two strings.
    int lcs(int i,int j,string str1,string str2){
            if(i<0||j<0){
                    return 0;
            }
            //match
            if(str1[i]==str2[j]){
                    return 1+lcs(i-1,j-1,str1,str2);
            }
            else{
                    return max(lcs(i-1,j,str1,str2),lcs(i,j-1,str1,str2));
            }

            //notmatch
}
    int shortestCommonSupersequence(string X, string Y, int m, int n)
    {
        int i=m-1;
  int j=n-1;
  return m+n-lcs(i,j,X,Y);
    }
};
```

Distinct Subsequences/Occurences:

```
class Solution
{
    public:
```

```
    int disticntOccurences(int i,int j,string s,string t){
        if(i<0&&j>=0){
            return 0;
        }
        if(i>=0&&j<0){
            return 1;
        }
        if(i<0&&j<0){
            return 1;
        }

        //match
        if(s[i]==t[j]){
            return
disticntOccurences(i-1,j-1,s,t)+disticntOccurences(i-1,j,s,t);
        }
        else{
            return disticntOccurences(i-1,j,s,t);
        }
    }
    int subsequenceCount(string s, string t)
    {
        int i=s.size()-1;
        int j=t.size()-1;
      return disticntOccurences(i,j,s,t);
    }
};
```