EASY :

1. Longest common prefix

```
class Solution {
public:
   string longestCommonPrefix(vector<string>& strs)
   {

       if(strs.size()==1)
       return strs[0];
       int k=0;
        string s="";

          string str=strs[0];
          if(str.size()==0)
          return "";
          for(int i=0;i<str.size();i++)
          {

             for(int j=1;j<strs.size();j++)
             {

                if(k>=strs[j].size()||str[i]!=strs[j][k])
                return s;

             }

          s+=str[i];
          k++;
          }

       return s;
   }
};
```

2. Min number of flips

```
int minFlips (string s)
{
   // your code here
   int n=s.size();
   if(n==1)
   return 0;
   else if(n==2 && s[0]==s[1])
```

```
    return 1;
    else if(n==2 && s[0]!=s[1])
    return 0;
    else
    {
        int c=0;
        int f=0;
        int f1=0;
        string s1="";
        string s2="";
        int c1=0;
        for(int i=0;i<n;i++)
        {
          if(i%2==0)
          {
              s1+='0';
              s2+='1';
          }
          else
          {
              s2+='0';
              s1+='1';
          }
        }
         for(int i=0;i<n;i++)
        {
           if(s[i]!=s1[i])
           c++;
           if(s[i]!=s2[i])
           c1++;
        }
       return min(c,c1);
    }
}
```

3. Second most repeated word in string

```
class Solution
{
  public:
    string secFrequent (string arr[], int n)
    {
        unordered_map<string,int>mp;
        int maxx=-1;
        int maxx2=-1;
        for(int i=0;i<n;i++)
```

```
        mp[arr[i]]++;

        for(auto it:mp)
        if(it.second>maxx)
        maxx=it.second;

        string ans="";
        for(auto it:mp)
        {
            if(it.second!=maxx and it.second>maxx2)
            {
                ans=it.first;
                maxx2=it.second;
            }
        }
        return ans;
    }
};
```

MEDIUM :

1. Reverse words in string

```
class Solution {
public:
    string reverseWords(string s)
    {
        stack<string>st;
        string ans;
        for (int i=0;i<s.size();i++)
        {
            string tmp;
            if(s[i]==' ')
            continue;
            while (i<s.size() && s[i]!=' ')
            {
                tmp+=s[i];
                i++;
            }
            st.push(tmp);
        }
        while (!st.empty())
        {
            ans+=st.top();
            st.pop();
```

```
            if(!st.empty())
            ans+= " ";
        }
        return ans;
    }
};
```

2. Integer to roman

```
class Solution {
public:

    string intToRoman(int num)
    {
    string str="";
    int arr[] = {1,4,5,9,10,40,50,90,100,400,500,900,1000};
    string arr1[] = {"I","IV","V","IX","X","XL","L","XC","C","CD","D","CM","M"};
    int i=12;
    while(num>0)
    {
        int rem=num/arr[i];
        num=num%arr[i];
        while(rem--)
        {
            str+=arr1[i];
        }
        i--;
    }
    return str;
    }
};
```

3. Find all anagrams in a string

```
class Solution {
public:
    vector<int> findAnagrams(string s, string p)
    {
        if(p.size()>s.size())
        return {};

        vector<int>res;
        int arr[26]={0};

        int n=p.size();
        for(int i=0;i<26;i++)
```

```
        arr[i]=0;
        for(int i=0;i<n;i++)
        arr[p[i]-'a']++;

        for(int i=0;i<=s.size()-n;i++)
        {
            int arr1[26]={0};
             for(int i=0;i<26;i++)
             arr1[i]=0;

            for(int j=i;j<i+n && j<s.size();j++)
            arr1[s[j]-'a']++;

            int f=0;
            for(int k=0;k<26;k++)
            {

                if(arr[k]!=arr1[k])
                {
                  f=1;
                  break;
                }
            }

            if(f==0)
            res.push_back(i);
        }
        return res;
    }
};
```

4. Compare version number

```
class Solution {
public:
    int compareVersion(string version1, string version2)
    {
        string s=version1;
        string s1=version2;
          vector<long long int>ans;
            vector<long long int>ans1;

            long long int sum=0;
         long long int sum1=0;
         long long int sum2=0;
```

```
        int n=s.size();
        int m=s1.size();

        for(int i=0;i<n;i++)
        {
           sum=0;
           while(s[i]!='.' && i<n)
           {
              sum+=(s[i]-'0');
              sum*=10;
              i++;
            }

           ans.push_back(sum);
        }
         for(int i=0;i<m;i++)
        {
           sum=0;
           while(s1[i]!='.' && i<m)
           {
             sum+=(s1[i]-'0');
             sum*=10;
              i++;
            }

           ans1.push_back(sum);
        }

      for(int i=0;i<ans.size();i++)
      sum1+=ans[i];
      for(int i=0;i<ans1.size();i++)
      sum2+=ans1[i];

      if(sum2==sum1)
      return 0;
      else if(sum1<sum2)
      return -1;
      else
      return 1;
    }
};
```

5. Min swaps for bracket balancing

```
class Solution{
public:
   int minimumNumberOfSwaps(string s)
   {
```

```
    int l=0;
    int r=0;


  int c=0;
  int diff= 0;

  for(int i=0;i<s.size();i++)
  {
     if(s[i]=='[')
     {
        l++;

        if(diff>0)
        {
             c+=diff;
             diff--;
        }
     }
     else if(s[i]==']')
     {
        r++;
        diff=(r-l);
     }
  }
  return c;
  }
};
```

6. Smallest window in a string containing all the characters of another string

```
class Solution
{
  public:
  string smallestWindow (string s, string p)
  {
     // Your code here
     int m=p.size();
     int n=s.size();
    if(m>n)
    return "-1";

    unordered_map<char,int> mp;
    for(char &c : p)
    mp[c]++;

    int c=mp.size();
```

```
        int i=0;
        int j=0;

        int start_Idx=-1;
        int win_Length=INT_MAX;

        while(j<n)
        {
            if(mp.find(s[j]) != mp.end())
            {
                mp[s[j]]--;

                if(mp[s[j]]==0)
                c--;
            }

            if(c==0)
            {
                while(c==0)
                {
                    if(mp.find(s[i]) != mp.end())
                    {
                        mp[s[i]]++;

                        if(mp[s[i]]==1)
                        c++;
                    }
                    i++;
                }

                if(win_Length > j-i+2)
                {
                    win_Length=j-i+2;
                    start_Idx=i-1;
                }
            }

            j++;
        }
        if(win_Length==INT_MAX)
        return "-1";
        else
        return s.substr(start_Idx, win_Length);
    }
};
```