# DAY 2 - STRING - **S.KHADEER BASHA**

1) **https://leetcode.com/problems/longest-substring-without-repeating-characters/**

```
class Solution {
 public:
  int lengthOfLongestSubstring(string s) {
    int ans = 0;
    vector<int> count(128);
    for (int l = 0, r = 0; r < s.length(); ++r) {
      ++count[s[r]];
      while (count[s[r]] > 1)
        --count[s[l++]];
      ans = max(ans, r - l + 1);
    }

    return ans;
  }
};
```

**2)https://leetcode.com/problems/minimum-remove-to-make-valid-parentheses/**

```
 class Solution {
 public:
  string minRemoveToMakeValid(string s)
    {
    stack<int> stack;  // unpaired '(' indices
    for (int i = 0; i < s.length(); ++i)
      if (s[i] == '(') {
        stack.push(i);  // Record the unpaired '(' index.
      } else if (s[i] == ')') {
        if (stack.empty())
          s[i] = '*';  // Mark the unpaired ')' as '*'.
        else
          stack.pop();  // Find a pair!
      }
    // Mark the unpaired '(' as '*'.
    while (!stack.empty())
      s[stack.top()] = '*', stack.pop();
    s.erase(remove(s.begin(), s.end(), '*'), s.end());
    return s;
  }
};
```

**3)[https://leetcode.com/problems/longest-palindromic-substring/](https://leetcode.com/problems/longest-palindromic-substring/)**

```
class Solution {
 public:
  string longestPalindrome(string s) {
    if (s.empty())
      return "";
    // (start, end) indices of the longest palindrome in s
    pair<int, int> indices{0, 0};

    for (int i = 0; i < s.length(); ++i) {
      const auto [l1, r1] = extend(s, i, i);
      if (r1 - l1 > indices.second - indices.first)
        indices = {l1, r1};
      if (i + 1 < s.length() && s[i] == s[i + 1]) {
        const auto [l2, r2] = extend(s, i, i + 1);
        if (r2 - l2 > indices.second - indices.first)
          indices = {l2, r2};
      }
    }
    return s.substr(indices.first, indices.second - indices.first + 1);
  }
 private:
  // Returns the (start, end) indices of the longest palindrome extended from
  // the substring s[i..j].
  pair<int, int> extend(const string& s, int i, int j) {
    for (; i >= 0 && j < s.length(); --i, ++j)
      if (s[i] != s[j])
        break;
    return {i + 1, j - 1};
  }
};
```

**4)[https://leetcode.com/problems/group-anagrams/](https://leetcode.com/problems/group-anagrams/)**

```
class Solution {
 public:
  vector<vector<string>> groupAnagrams(vector<string>& strs) {
    vector<vector<string>> ans;
    unordered_map<string, vector<string>> keyToAnagrams;
    for (const string& str : strs)
   {
      string key = str;
      ranges::sort(key);
      keyToAnagrams[key].push_back(str);
    }

    for (const auto& [_, anagrams] : keyToAnagrams)
      ans.push_back(anagrams);
    return ans;
  }
};
```

**5)[https://leetcode.com/problems/generate-parentheses/](https://leetcode.com/problems/generate-parentheses/)**

```
class Solution {
 public:
  vector<string> generateParenthesis(int n) {
    vector<string> ans;
    dfs(n, n, "", ans);
    return ans;
  }
 private:
  void dfs(int l, int r, string&& path, vector<string>& ans) {
    if (l == 0 && r == 0) {
      ans.push_back(path);
      return;
    }
    if (l > 0) {
      path.push_back('(');
      dfs(l - 1, r, move(path), ans);
      path.pop_back();
    }
    if (l < r) {
      path.push_back(')');
      dfs(l, r - 1, move(path), ans);
      path.pop_back();
    }
  }
};
```

**6)[https://leetcode.com/problems/basic-calculator-ii/](https://leetcode.com/problems/basic-calculator-ii/)**

```
class Solution {
 public:
  int calculate(string s) {
    int ans = 0;
    int prevNum = 0;
    int currNum = 0;
    char op = '+';
    for (int i = 0; i < s.length(); ++i) {
      const char c = s[i];
      if (isdigit(c))
        currNum = currNum * 10 + (c - '0');
      if (!isdigit(c) && !isspace(c) || i == s.length() - 1) {
        if (op == '+' || op == '-') {
          ans += prevNum;
          prevNum = op == '+' ? currNum : -currNum;
        } else if (op == '*') {
          prevNum *= currNum;
        } else if (op == '/') {
          prevNum /= currNum;
        }
        op = c;
        currNum = 0;
      }
    }

    return ans + prevNum;
  }
};
```

**7)https://leetcode.com/problems/integer-to-roman/**

```
class Solution
{
 public:
  string intToRoman(int num)
 {
   const vector<pair<int, string>> valueSymbols
   {
       {1000, "M"}, {900, "CM"}, {500, "D"}, {400, "CD"}, {100, "C"},
       {90, "XC"},  {50, "L"},   {40, "XL"}, {10, "X"},   {9, "IX"},
       {5, "V"},    {4, "IV"},   {1, "I"}
   };
   string ans;
   for (const auto& [value, symbol] : valueSymbols)
   {
     if (num == 0)
       break;
     while (num >= value)
     {
       num -= value;
       ans += symbol;
     }
   }
   return ans;
 }
};
```

**8)[https://leetcode.com/problems/reverse-words-in-a-string/](https://leetcode.com/problems/reverse-words-in-a-string/)**

**PYTHON CODE :**

```
class Solution:
  def reverseWords(self, s: str) -> str:
    return ' '.join(reversed(s.split()))
```

**9)[https://leetcode.com/problems/simplify-path/](https://leetcode.com/problems/simplify-path/)**

```
class Solution
```

```
{
 public:
  string simplifyPath(string path)
  {
    string ans;
    istringstream iss(path);
    vector<string> stack;

    for (string dir; getline(iss, dir, '/');)
    {
      if (dir.empty() || dir == ".")
        continue;
      if (dir == "..") {
        if (!stack.empty())
          stack.pop_back();
    }
      else
      {
        stack.push_back(dir);
      }
    }
    for (const string& s : stack)
      ans += "/" + s;

    return ans.empty() ? "/" : ans;
  }
};
```

**10)**[**https://leetcode.com/problems/zigzag-conversion/**](https://leetcode.com/problems/zigzag-conversion/)

```
class Solution
```

```
{
 public:
  string convert(string s, int numRows)
  {
    string ans;
    vector<vector<char>> rows(numRows);
    int k = 0;
    int direction = (numRows == 1) - 1;
    for (const char c : s)
    {
      rows[k].push_back(c);
      if (k == 0 || k == numRows - 1)
        direction *= -1;
      k += direction;
    }
    for (const vector<char>& row : rows)
      for (const char c : row)
        ans += c;
    return ans;
  }};
```