Greedy Algorithms
(Assignment Solutions)

Question 1 :

```
    int balancedStringSplit(string s) {
              // The counter to keep track of how many possible balance split
string can do
             int ans = 0;
              // The R Counter to keep track the number of R in the current
split string
             int countR = 0;
              // The L Counter to keep track the number of L in the current
split string
             int countL = 0;
             //   Start   traversing   the   string
             for(size_t i = 0; i < s.length(); i++) {
                 // If the current character is R or L, we increment the R
counter or L counter (Remember we need to keep track of the number of R and L
in the current split string)
                 if(s[i] == 'R') {
                     countR++;
                 } else if (s[i] == 'L') {
                     countL++;
                 }
                  // Once the string is balanced, "split" it (increase ans
counter to indicate we split it, then reset the R and L counters to 0 for new
split string)
                 if(countR == countL) {
                     ans++;
                     countR = 0;
                     countL = 0;
                 }
             }
             // Return the count of string we can split
             return ans;
         }
```

Question 2 :

```
string largestOddNumber(string num) {
       for (int i=num.length()-1; i>=0; i--){
           int digit = num[i]-'0' ;
           if (digit & 1){
               return num.substr(0, i+1) ;
           }
       }
       return "" ;
   }
```

Question 3 :

```
string getSmallestString(int n, int k) {
       string ans;
       while(n!=0){
           int ch = k-(n-1)*26;
           if(ch<=0){
               ans.push_back('a');
               k--;
               n--;
           }
           else{
               ans.push_back(ch+'a'-1);
               n--;
               k -= (ch);
           }
       }
       return ans;
   }
```

Question 4 :

```
public int maxProfit(int[] prices) {
       int profit = 0;
       int n = prices.length;

       int max[] = new int[prices.length];
       max[n-1] = prices[n-1];

       for(int i=n-2; i>=0; i--) {
           max[i] = Math.max(max[i+1], prices[i]);
       }

       for(int i=0; i<n; i++) {
           int currProfit = max[i] - prices[i];
           profit = Math.max(currProfit, profit);
       }

       return profit;
   }
```

Question 5 :

```
int splitArray(vector<int>& nums, int k) {
       long long int mn = INT_MIN;
       long long int mid, ans, mx = 0, sum;

       int tmp;
       for(auto &i: nums){
           mx += i;//Max possible sum of subarray
           mn = max(mn,i*1LL);//Max of minimum possible sum of subarray
       }
       while(mn<=mx){
           mid = (mx-mn)/2+mn;
           tmp = 1,sum=0;
           for(auto &i: nums){
               sum += i;
```

```
                if(sum>mid){
                    tmp++;
                    sum = i;
                }
            }
            if(tmp==k){
                ans = mid;
                mx = mid-1;
            }else if(tmp>k){
                mn = mid+1;
            }else{
                ans = mid;
                mx = mid-1;
            }
        }
        return ans;
    }
```