LeetCode Problem Solving

Week-3

Q. Check Divisibility by Digit Sum & Product → Number 3622

=> Given an number. We have to find wether it's divisible or not by it's numbers sum and product sum. Example: Number 23. So it's numbers sum and product are 2+3=5 & 2*3=6 so 5+6=11. now 23 divisible by 11? now because 23%11!=0. That's why we can say 23 is not divisible. Here is the code:

```
int multiply(int n){
        int temp,mul,rem;
        while(n!=0){
                temp = n;
                rem = temp%10;
                n = n/10;
                mul = rem*multiply(n);
                return mul;
        }
        return 1;
}

int sum(int n){
        int temp,sums,rem;
        while(n!=0){
                temp = n;
                rem = temp%10;
                n = n/10;
                sums = rem+sum(n);
                return sums;
        }
        return 0;
}

bool checkDivisibility(int n) {
        int sums,mul,res;
        mul = multiply(n);
        sums = sum(n);
        res = mul+sums;
        if(n%res==0){
                return true;
        }
        else{
                return false;
        }
}
```

Q. Valid Palindrome → Number 125

=> Given a string. Our target is to say wether this string palindrome or not. Palindrome is we will take a string only lower cases letter or digit. Then we will reverse it. If after reverse It remain same then its palindrome otherwise not. Here is the code:

```
bool isPalindrome(char* s) {
        int i,j,k;
        char* p;
        char* q;
        bool result;
        k = 0;
        p = (char*)malloc(strlen(s)+1);
        q = (char*)malloc(strlen(s)+1);
        for(i=0;s[i]!='\0';i++){
                if(s[i]>='A' && s[i]<='Z'){
                        p[k++] = s[i]+32;
                }
                else if((s[i]>='a' && s[i]<='z') || (s[i]>=48 && s[i]<=57)){
                        p[k++] = s[i];
                } else{
                        continue;
                }
        }
        p[k]='\0';
        for(j=0;j<k;j++){
                q[j] = p[k-1-j];
        }
        q[k]='\0';
        if(strcmp(p,q)==0){
                result = true;
        } else{
                result = false;
        }
        free(p);
        free(q);
        return result;
}
```

Here is more furnished and cleaner code

```
bool isPalindrome(char* s) {
        int i,j,k=0,len=strlen(s);
        char* p = (char*)malloc(len+1);
        char* q = (char*)malloc(len+1);
        bool result = false;
        for(i=0;s[i]!='\0';i++){
                if(isalnum(s[i])){
                        p[k++] = tolower(s[i]);
                }
        }
        p[k]='\0';
        for(j=0;j<k;j++){
                q[j] = p[k-1-j];
        }
        q[k]='\0';
        if(strcmp(p,q)==0){
                result = true;
        }
        free(p);
        free(q);
        return result;
}
```

Q. Majority Element → Number 169

=> Given an array. Our target is to find which number comes most of the time. We can solve this using Voting Algorithm. In this algorithm we will use two variable one for count and one for candidate. We will do a loop from 0 to until numsSize. Then in loop we will check either count 0 or not. If count 0 then we will choose current number as candidate and increase count value. Then we will check the current number either its equal to candidate or not. If its candidate then count value will increase otherwise count will decrease. After loop simply return candidate.

```
int majorityElement(int* nums, int numsSize) {
        int count = 0, candidate = 0;
        for (int i = 0; i < numsSize; i++) {
                if (count == 0) {
                        candidate = nums[i];
                        count = 1;
                } else if (nums[i] == candidate) {
                        count++;
                } else {
                        count--;
                }
        }
        return candidate;
}
```