Assignment 1:

1. Two Sum
   Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

   Code:

```cpp
//Time Complexity: O(N^2)
//Space Complexity: O(N)
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> r;
        for(int i=0;i<nums.size();i++)
        {
            for(int j=i+1;j<nums.size();j++)
            {
                if(nums[i]+nums[j]==target)
                {
                    r={i,j};
                }
            }
        }
        return r;

    }
};
```

2. Remove Element
   Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

```cpp
//Time Complexity: O(N)
//Space Complexity: O(1)
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int count=0;
        for(int i=0;i<nums.size();i++)
        {
            if(nums[i]!=val)
            {
               nums[count++]=nums[i];

            }
        }

        return count;

    }

};
```

3. Search Insert Position
   Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order
   Code:

```
//Time Complexity: O(log n)
//Space Complexity:O(1)
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
    int low=0, high=nums.size()-1, mid;
    if(target<nums[low])
    {
        return 0;
        //cout<<nums[left];
    }
     if(target>nums[high])
    {
        return nums.size();
        //cout<<nums[right];
    }
    while(low<=high)
    {
        mid=low+(high-low)/2;
        if(target>nums[mid])
        {
            low=mid+1;
            cout<<low;
        }
        else if(target<nums[mid])
        {
            high=mid-1;
            cout<<high;
        }
        else
        {
            return mid;
            cout<<mid;
        }
    }
    return low;
    cout<<low;
    }};
```

4. Plus One
   You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

   Code:

```cpp
//Time Complexity: O(N)
//Space Complexity:O(1)
class Solution {
public:
    vector<int> plusOne(vector<int>& digits) {
        for(int i=digits.size()-1; i>=0; i--)
        {
            if (digits[i]<9)
            {
                digits[i]++;
                return digits;
            }
            else
            {
                digits[i]=0;
            }
        }
        digits.insert(begin(digits), 1);
        return digits;
    }
};
```

5. Merge Sorted Array

You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

Code:

```
//Time Complexity: O(N)
//Space Complexity: O(1)
class Solution
{
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n)
    {
    int i=m-1;
    int j=n-1;
    int k=m+n-1;
        while(i>=0 && j>=0)
        {
            if(nums1[i]>nums2[j])
            {
                nums1[k]=nums1[i];
                i--;
                k--;
            }
            else
            {
                nums1[k]=nums2[j];
                k--;
                j--;
            }
        }
        while(i>=0)
        {
            nums1[k]=nums1[i];
            k--;
            i--;
        }
         while(j>=0)
        {
            nums1[k]=nums2[j];
            k--;
            j--;
        }
```

```
        }
    };
```