www.mmrkhan.webs.com
khan4019@gmail.com

Table of Contents

Tricky stuff:
Remove duplicate char
Greatest common divisor

1. Numbers

1.1 binary number generator

- Can't take 0 as input

```
private static void binaryConversion(int input)
      {
         Stack<int> bin = new Stack<int>();
         while (input >0)
         {
            bin.Push(input % 2);
            input /= 2;
         }
         string s=string.Empty;
         int len = bin.Count;
         for (int i = 0; i < len; i++)
            s += bin.Pop();
         Console.Write(s);
```

1.2 Finding all prime factors (can u make it better!)

- Worst case: O(n/2)
- You could take benefit of the division from 3, there would not be any even. Hence comlextity could be O(n/2)

```
private static void primeFactor(int input)
      {
         if(input<=2)
            Console.WriteLine("Please enter a positive number more than 2");
         int divisor=2;
         while(input>=2)
         {
            if (input % divisor == 0)
```

```
            {
                Console.Write(divisor + " ");
                input /= divisor;
            }
            else if (divisor != 2)
                divisor+=2;

            else
                divisor++;
        }
```

1.3 check whether a number is a prime or not

- I know why not more than n/2: but not sure about sqrt(n)
- Think like 3 diye check korle Jodi divided na hoi…then one-third bigger number diye divided hobe na
- For example: 35 doesn't divided by 3 then it will not be dividable by number more than (35/3)..hence you don't have to check numbers between 35-35/3…take your time and try to understand
- Say u have 127…don't go by 2 hence if it is divisible by anything…that should be less than 63.5>>>doesn't go by 3…so, divisor must be less then 127/3=42>>don't go by 5, divisor should be less than 127/5>>not by 7, should be less than 127/7….if u go this way…than it comes like 11 and 11…..use ur brain

```
private static bool checkPrime(int input)
    {
        bool isPrime=true;
        if (input == 1 || input == 2)
            isPrime=true;
        if (input % 2 == 0 && input != 2)
            isPrime=false;
        else
        {
            int divisor = 3;
            double k = Math.Sqrt(input);
            while (divisor <= k && isPrime)
            {
                if (input % divisor == 0)
                    isPrime = false;
                else
                    divisor += 2;
            }
        }
        return isPrime;
    }
```

1.4 Greatest common divisor

- It's not as simple, as it look like…try to get the concept to explain during the interview

```
private static int greatestCommonDivisor(int a, int b)
    {
        if (b == 0)
            return a;
        else
            return greatestCommonDivisor(b, a % b);
    }
```

www.mmrkhan.webs.com
khan4019@gmail.com

1.5 Factorial by recursion

```
static int FactbyRecursion(int n)
      {
         if (n== 1)
            return 1;
         else
            return FactbyRecursion(n - 1) * n;
      }
```

1.6 Factorial with intermediate results

```
private static int[] FactWithIntermediate(int n)
      {
         int[] intermediateFactorial = new int[n];
         int factorial = 1;
         for 0028int i = 1; i <= n; i++)
         {
            factorial *= i;
            intermediateFactorial[i - 1] = factorial;
         }
         return intermediateFactorial;
      }
```

1.7 Convert String to a number

```
private static int stringToNumb(string input)
      {
         char[] charArray = input.ToCharArray();
         int number = 0;
         int negative=1;
         int start = 0;
         if (charArray[0] == '-')
         {
            negative = -1;
            start = 1;
         }
         for (int i = start; i < input.Length; i++)
         {
            if (charArray[i] < '0' || charArray[i] > '9')
               return -1;
            else
               number = number * 10 + charArray[i] - '0';
         }
         return number*negative;
      }
```

1.8 Convert number to String

- Be careful to use it. A lot of things to be very careful.
- Very poor way of coding…though its working

```
private static string numberToString(int n)
      {
         int[] def = new int[15];
         bool negative=false;
         string output =string.Empty;
         if(n<0)
         {
            negative=true;
            n*=-1;
         }
         int i = 0;
         while(n>0)
         {
            def[i] = '0'+n%10;
            i++;
            n /= 10;
         }
         if (negative)
         {
            def[i++] = '-';
         }
         while(i>0)
         {
            output += char.ConvertFromUtf32(def[i-1]);
            i--;
         }
         return output;
      }
```

1.9 shortened a number by repeating numbers

- First problem was not showing ending number (1 exceptional case)
- Then not showing if the ending number is only once
- Pretty messy code…need to modify it

```
private static string ShorteneStringWithNumber(string input)
      {
         char[] inputToArray = input.ToCharArray();
         int[] outputArray = new int[inputToArray.Length * 2];
         string output= string.Empty;
         int numberCount = 1;
         int typeCount = 0;
         for (int i = 1; i < inputToArray.Length; i++)
         {
            if ((inputToArray[i] == inputToArray[i - 1])&&(i!=inputToArray.Length-1))
               numberCount++;
            else if (i != inputToArray.Length - 1)
            {
               outputArray[typeCount] = inputToArray[i - 1];
               outputArray[typeCount + 1] = numberCount;
               numberCount = 1;
               typeCount += 2;
            }
            else if ((i == inputToArray.Length - 1) && (inputToArray[i] == inputToArray[i - 1]))
            {
               outputArray[typeCount] = inputToArray[i - 1];
```

```
            outputArray[typeCount + 1] = numberCount + 1;
        }
        else if ((i == inputToArray.Length - 1) && (inputToArray[i] != inputToArray[i - 1]))
        {
            outputArray[typeCount+1] = inputToArray[i];
            outputArray[typeCount + 2] = 1;
        }
    }
    foreach (int k in outputArray)
    {
        if (k > 9)
            output += char.ConvertFromUtf32(k);
        else if (k > 0)
            output += k;
    }
    return output;
}
```

1.10 displays number of Prime numbers 1□N (inclusive) (less efficient)

- Not that efficient but little better than try to divide by all numbers or all odd numbers. Trying to divide by prime numbers excluding 2 (even numbers are not assigned to be checked)
- Careful about initial value of the variables
- Number of prime numbers could not be more than n/2. However, if u enter 3, then no of prime numbers would be (n=1)/2
- Since, until 3 is automatically inserted. Prime check starts from 5

```
private static int[] displayPrimeNumbers(int n)
    {
        int[] PrimeNumbers = new int[(n+1) / 2];
        int primeCounter=2;
        int prime=5;
        bool isPrime=true;
        if (n >= 3)
        {
            PrimeNumbers[0] = 2;
            PrimeNumbers[1] = 3;
        }
        else if (n == 2)
            PrimeNumbers[0] = 2;
        while (prime <= n)
        {
            for (int i = 1; i < primeCounter; i++)
            {
                if (prime % PrimeNumbers[i] == 0)
                {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime)
            {
                PrimeNumbers[primeCounter] = prime;
                primeCounter++;
            }
```

```
            prime += 2;
            isPrime = true;
        }
        return PrimeNumbers;
    }
```

1.11 displays number of Prime numbers 1□N (inclusive) (efficient)

- Just changed one line: while checking prime numbers only divide by the prime numbers lesser than the sqrt of the number:::: I know this need a lot of thought to believe that this will work
- Can it be better?

```
private static int[] displayPrimeNumbers(int n)
    {
        int[] PrimeNumbers = new int[(n+1) / 2];
        int primeCounter=2;
        int prime=5;
        bool isPrime=true;
        if (n >= 3)
        {
            PrimeNumbers[0] = 2;
            PrimeNumbers[1] = 3;
        }
        else if (n == 2)
            PrimeNumbers[0] = 2;
        while (prime <= n)
        {
            for (int i = 1; i < primeCounter && PrimeNumbers[i]*PrimeNumbers[i]<=prime; i++)
            {
                if (prime % PrimeNumbers[i] == 0)
                {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime)
            {
                PrimeNumbers[primeCounter] = prime;
                primeCounter++;
            }
            prime += 2;
            isPrime = true;
        }
        return PrimeNumbers;
    }
```

1.12 Displays number of days in a month

- Baby code…

```
private static int numberOfDaysInMonth(int month, int year)
    {
```

```
        if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month ==
12)
            return 31;
        else if (month == 4 || month == 6 || month == 9 || month == 11)
            return 30;
        else if (month == 2 && year % 4 != 0)
            return 28;
        else
            return 29;
    }
```

1.13 Display number divisible by 15 (could be 5,3,4)

- Baby code:
- Look how two variables used in one single loop
- Time complexity O(n)
- Start i at 15 or the divisor
- Look the size of the output array
- Write test cases in your own: range coverage

```
private static int[] numberDivisibleBy15(int n)
    {
        int[] outputArray = new int[n / 15];
        for(int i=15,j=0;i<=n && j<=n/15;i+=15, j++)
            outputArray[j]=i;
        return outputArray;
    }
```

1.14 if divisible by 3 display “Baya” if 5 display “Soft” if by both display “BayaSoft” else number in new line

```
 private static void printNumbers(int n)
  {
    if (n < 1)
        Console.WriteLine("Please give a number bigger than 1");
    for (int i = 1; i <= n; i++)
    {
        if (i % 15 == 0)
            Console.WriteLine("BayaSoft");
        else if (i % 3 == 0)
            Console.WriteLine("Baya");
        else if (i % 5 == 0)
            Console.WriteLine("Soft");
        else
            Console.WriteLine(i);
    }
  }
```

1.15 Swap two numbers without temporary variable (add or deduct)

```
private static void swapWithoutTemp(int a, int b)
    {
        Console.WriteLine("a:" + a);
```

```
        Console.WriteLine("b:" + b);
        b = b - a;
        a = a + b;
        b = a - b;
        Console.WriteLine("a:" + a);
        Console.WriteLine("b:" + b);
    }
```

1.16 Swap two numbers without temporary variables (bit manipulation)

```
private static void swapValues(int a, int b)
    {
        Console.WriteLine("a: " + a + " and b: " + b);
        a = a ^ b;
        b = a ^ b;
        a = a ^ b;
        Console.WriteLine("a: " + a + " and b: " + b);
    }
```

1.17 computes the number of trailing zeros in n factorial (tricky)

- Question is tricky for me: ask for last zeros in factorial like 6!=720 trailing zero=1
- Look into the for loop:
- Knowledge: trailing zeros are contributed by pair of 5 and 2 because 5*2=10. To count the number of pairs, we just have to count the number of multiples of 5. Note that while 5 contributes to one multiple of 10, 25 contributes two (because 5*5=25)

```
private static int missingNumber(int[] inputArray, int maxNumber)
    {
        int sum = maxNumber * (maxNumber + 1) / 2;
        int arraySum=0;
        for (int i = 0; i < inputArray.Length; i++)
            arraySum += inputArray[i];
        return sum - arraySum;
    }
```

1.18 max of two numbers without if else (tricky and tough)

- Style 1: if a>b, return a else b
- Style 2: if (a-b) is negative return b else return a
- Style 3: if (a-b) is negative, let k=1 else k=0. Return a-k*(a-b)
- Style 4: let c=a-b. let k = most significant bit of c. return a-k*c
- Try to get it

```
private static int MaxOfTwo(int a, int b)
    {
        int c = a - b;
        int k = (c >> 31) & 0X1;
        int max = a - k * c;
        return max;
    }
```

www.mmrkhan.webs.com
khan4019@gmail.com

1.19 nth Fibonacci Number (recursive)

- Iterative you should be able to do yourself

```
private static int nthFibonacci(int n)
      {
         if (n <= 1)
            return n;
         else
            return nthFibonacci(n - 1) + nthFibonacci(n - 2);
      }
```

1.20 Random 1 to 7 from function random 1 to 5

- Try to get how this works

```
private static int rand5()
      {
         Random rand = new Random();
         int random = rand.Next(1, 5);
         return random;
      }
      private static int rand7()
      {
         int num = 5 * (rand5() - 1) + (rand5() - 1);
         if (num < 21)
            return (num % 7 + 1);
         else
            return rand5();
      }
```

1.21 missing number (ask Minhaz vai about this problem)

- Careful about formula…I have all

```
private static int missingNumber(int[] inputArray, int maxNumber)
      {
         int sum = maxNumber * (maxNumber + 1) / 2;
         int arraySum=0;
         for (int i = 0; i < inputArray.Length; i++)
            arraySum += inputArray[i];
         return sum - arraySum;
      }
```

www.mmrkhan.webs.com
khan4019@gmail.com

2. Array

2.1 Remove Duplicate number from unsorted Array

- Time Complexity: $O(n^2)$
- Require Extra Memory // I used an array
- O(n): have to range…then array
- Alternative: sort the array

```
static void removDupNum(int[] a)
      {
         int[] b=new int[a.Length];
         int length= a.Length;
         bool duplicate;
         int bPoint=1;
         b[0]=a[0];
         for (int i = 1; i < length; i++)
         {
            duplicate = false;
            for (int j = 0; j <bPoint; j++)
            {
               if (a[i] == b[j])
               {
                  duplicate = true;
                  break;
               }
            }
            if (!duplicate)
            {
               b[bPoint] = a[i];
               bPoint++;
            }
         }
         for (int k=0;k<bPoint;k++)
            Console.Write(b[k] + " ");
      }
```

2.2 Remove duplicate item from a sorted array and then find its length

- Its ok..normal thing

```
private static int[] removeDuplicate(int[] inputArray)
      {
         int[] outputArray = new int[inputArray.Length];
         outputArray[0] = inputArray[0];
         int outputCount=1;
         for (int i = 1; i < inputArray.Length; i++)
         {
            if (inputArray[i] != inputArray[i - 1])
            {
               outputArray[outputCount] = inputArray[i];
               outputCount++;
            }
         }
```

```
        return outputArray;
    }
```

2.2 Merge two sorted array

- Complexity: O(n). n means summation of the elements in the both array
- Remember: difference in length. What happen one have been inserted.
- Loop: I used for loop, u can use while
- Can't be better than this!!

```
static void mergeSortedArray(int[] a, int[] b)
    {
        int alen = a.Length;
        int blen = b.Length;
        int[] mergedArray = new int[alen + blen];
        int cycle;
        if (alen < blen)
            cycle = alen;
        else
            cycle = blen;
        int aPoint = 0;
        int bPoint = 0;

        for (int i = 0; i < alen + blen;i++ )
        {
            if (aPoint == alen)
            {
                mergedArray[i] = b[bPoint];
                bPoint++;
            }
            else if (bPoint == blen)
            {
                mergedArray[i] = a[aPoint];
                aPoint++;
            }
            else if (a[aPoint] <= b[bPoint])
            {
                mergedArray[i] = a[aPoint];
                aPoint++;
            }
            else
            {
                mergedArray[i] = b[bPoint];
                bPoint++;
            }
        }

        foreach (int c in mergedArray)
            Console.Write(c + " ");
    }
```

www.mmrkhan.webs.com
khan4019@gmail.com

2.3 Remove Duplicate number from sorted array

- Memory: use of extra memory
- Time Complexity: O(n)

```
static int[] removDupNum(int[] a)
      {
         int[] b = new int[a.Length];
         b[0]=a[0];
         int bPointer=1;
         for (int i = 1; i < a.Length; i++)
         {
            if (a[i] != b[bPointer - 1])
            {
               b[bPointer] = a[i];
               bPointer++;
            }
         }
         return b;
      }
```

2.4 Product of top 3 numbers (seems easy but tricky)

- Bug: can integer have negative number: if not, put second , third to zero: discuss bug if three negative numbers
- Think about and talk about exceptional cases
- If yes, u r in trouble…boy: discuss bug in {2,-1,-12} and set second and third to be lowest in the data type.
- Be careful with your tricks…

```
private static int top3NumberMultiplication(int[] input)
      {
         int length = input.Length;
         if (length < 3)
            return -1;
         int top=input[0];
         int second = int.MinValue;
         int third = int.MinValue;

         for (int i=1;i<length;i++)
         {
            if (input[i] > top)
            {
               third = second;
               second = top;
               top = input[i];
            }
            else if (input[i] > second)
            {
               third = second;
               second = input[i];
            }
            else if (input[i] > third)
               third = input[i];
         }
         return top * second * third;
```

```
}
```

2.5 Swap deck of card without conflict

- Generate a random number choose the card and put it to be the last before the previously set
- Now generate random number between 0 to 52-1-i

```
private static int[] swapDeck(int[] inputDeck)
    {
      Random rand = new Random();
      for (int i = 0; i < 52; i++)
      {
        int n = rand.Next(0, 52-1-i);
        int temp = inputDeck[52 - 1 - i];
        inputDeck[52 - 1 - i] = inputDeck[n];
        inputDeck[n] = temp;
      }
      return inputDeck;
    }
```

2.6 display numbers which are repeated even times in an array of number

- Does it contains only int? is it sorted? What is the size is it very big?
- Assumptions: giving sorted array>> at least repetition is seated next to each other
- Problem: loop>>start from 1 and compare backward::make sure it counts the last one
- Trick: length of the output array could be maximum of half of the length of the input array..because if everyone repeats for minimum of two time…that's the worst case.
- Carefully: reset counter=1 {why to one not to 0?}, increase output array pointer, catch exceptional cases

```
private static int[] evenNumberOfRepeatation(int[] inputArray)
    {
      int count =1;
      int[] outputArray = new int[inputArray.Length / 2];
      int outputCounter=0;

      for (int i = 1; i < inputArray.Length; i++)
      {
        if ((inputArray[i] == inputArray[i - 1]) && (i != inputArray.Length - 1))
          count++;
        else if ((count % 2 == 0) && (i != inputArray.Length - 1))
        {
          outputArray[outputCounter] = inputArray[i - 1];
          count = 1;
          outputCounter++;
        }
        else if ((i == inputArray.Length - 1) && (++count % 2 == 0))
          outputArray[outputCounter] = inputArray[i];
      }
      return outputArray;
    }
```

2.7 Maximum product of two consecutive numbers in an array

```
private static int maxConsecutiveProduct(int[] inputArray)
    {
       int maxProduct = 0;
       for (int i = 1; i < inputArray.Length; i++)
       {
          if (inputArray[i] * inputArray[i - 1] > maxProduct)
             maxProduct = inputArray[i] * inputArray[i - 1];
       }
       return maxProduct;
    }
```

2.8 Consecutive sequence with largest sum (return the sum)

- Try to understand: consecutive sequence with largest sum
- Hence, if sum drops below zero, that's one is going away
- It has flaw: if all the numbers are negative: what will u day (use your own brain)

```
private static int MaxContinuousSum(int[] inputArray)
    {
       int maxSum = 0;
       int sum=0;
       for (int i = 0; i < inputArray.Length; i++)
       {
          sum += inputArray[i];
          if (sum > maxSum)
             maxSum = sum;
          else if (sum < 0)
             sum = 0;
       }
       return maxSum;
    }
```

2.9 find all pairs in an array which sum to a specific value (unsorted array)

3. String Manipulation

3.1 Find the first non-repeating character (case sensitive)

- This is case sensitive…I mean "M" and m are different: must talk about that during interview

```
private static void firstNonRepeatingChar(string input)
    {
      char[] inputArray = input.ToCharArray();
      int[] ascii = new int[256];
      for (int i = 0; i < inputArray.Length; i++)
      {
        ascii[inputArray[i]] += 1;
      }
      for (int j = 0; j < inputArray.Length; j++)
      {
        if (ascii[inputArray[j]] == 1)
        {
          Console.Write(inputArray[j] + " ");
          break;
        }
      }
    }
```

3.2 Find the first non-repeating character (case insensitive)

- Very small trick to make it case insensitive: find out yourself

```
private static void firstNonRepeatingChar(string input)
    {
      char[] inputArray = input.ToCharArray();
      int[] ascii = new int[256];
      for (int i = 0; i < inputArray.Length; i++)
      {
        if (inputArray[i]<='Z')
        ascii[inputArray[i]+32] += 1;
        else
          ascii[inputArray[i]] += 1;
      }
      for (int j = 0; j < inputArray.Length; j++)
      {
        if (ascii[inputArray[j]] == 1)
        {
          Console.Write(inputArray[j] + " ");
          break;
        }
      }
    }
```

3.3 char count in a string (count nonwhite space)

- I don't C# allow to access it and count it…how strange
- Exact as Microsoft word
- U don't have to convert a string to char array

```
private static int charCount(string input)
    {
      int count=0;
```

```
        for (int i = 0; i < input.Length; i++)
        {
            if (input[i] != ' ')
                count++;
        }
        return count;
    }
```

3.4 word count in a string

- Exactly same as Microsoft word
- Just for last word…need to put an exceptional case
- **Remember exceptional cases:** multiple white spaces, string started with white space, string ends with white space, counts based only white space not by '.' Or meaning of word.
- Works for multiple spaces between words (get the concept yourself)

```
private static int stringCount(string input)
    {
        int wordCount=0;
        for (int i = 1; i < input.Length; i++)
        {
            if (input[i - 1] != ' ' && input[i] == ' ')
                wordCount++;
            else if (i == input.Length - 1 && input[i] != ' ')
                wordCount++;
        }
        return wordCount;
    }
}
```

3.5 Remove Duplicate Character from a String

- Complexity: O(n)
- Be careful. Its little tricky
- Use extra memory
- Use library function
- Concept: if not find the char in the index will return -1

```
static string RemovDupChar(string s)
    {
        string result = "";
        foreach (char c in s)
        {
            if (result.IndexOf(c) == -1)
                result += c;
        }
        return result;
    }
```

www.mmrkhan.webs.com
khan4019@gmail.com

3.6.1 Reverse a string general way (iterative)

- Careful: **len-1-i**
- Trivial one…average performance and overhead of copying twice
- One to char array and other to original as strings are immutable
- **Amar loop e problem ase…odd number len hoile cholbe na:: (len+1)/2**

```
private static string ReverseStringWithSwap(string s)
    {
      char[] str = s.ToCharArray();
      int len=str.Length;
      for (int i = 0; i <(len+1)/2; i++)
      {
        char temp = str[i];
        str[i] = str[len -1- i];
        str[len-1 - i]=temp;
      }
      return new string(str);
    }
```

3.6.2 Reverse a string without temp but converting to char Array

- **Amar loop e problem ase…odd number len hoile cholbe na**

```
private static string ReverseWithoutTemp(string s)
    {
      char[] str = s.ToCharArray();
      int len = str.Length;
      for (int i = 0; i < (len+1) / 2; i++)
      {
        str[i] = s[len - 1-i];
        str[len-1-i]=s[i];
      }
      return new string(str);
    }
```

3.6.3 Reverse a string with Recursive way

- Somehow it works: I don't know how

```
private static string ReverseWithRecursion(string s, int len)
{
  if (len == 1)
    return s;
  else
    return ReverseWithRecursion(s.Substring(1, s.Length - 1), --len) + s[0].ToString();
}
```

3.6.4 Reverse a String with Stack (fun fun fun)

- Two loops is used: performance is almost equal to the normal reversal
- Be careful about string declaration and pop method…it's not easy for u baby!!!

```
private static string reverseWithStack(string s)
    {
      Stack<string> inputStack = new Stack<string>();

      for (int i = 0; i < s.Length; i++)
        inputStack.Push(s.Substring(i, 1));
      string revString = string.Empty;
      for (int i = 0; i < s.Length; i++)
        revString += inputStack.Pop();
      return revString;
    }
```

3.6.5 Reverse a String without char array

- Be careful with your loop

```
private static string ReverseStringWithoutCharArray(string s)
    {
      int len=s.Length;
      char[] revStr = new char[len];

      for (int i = 0, j = len - 1; i <= j;i++,j-- )
      {
        revStr[i] = s[j];
        revStr[j] = s[i];
      }
      return new string(revStr);
    }
```

3.6.6 Reverse String by using Bit Manipulation XOR

- No idea how it works: don't even try at interview

```
private static string ReverseWithBitXOR(string s)
    {
      char[] str = s.ToCharArray();
      int len = s.Length - 1;
      for (int i = 0; i < len; i++, len--)
      {
        str[i] ^= str[len];
        str[len] ^= str[i];
        str[i] ^= str[len];
      }
      return new string(str);
    }
```

3.7 Reverse Word and Append in String Builder

```
private static string revWords(string inputString)
    {
      // using string builder
      StringBuilder sb = new StringBuilder();
```

```
        string[] wordArray = inputString.Split(' ');
        Array.Reverse(wordArray);
        string revWord= string.Empty;
        foreach (string s in wordArray)
            sb.Append(" "+s);
        return sb.ToString();
    }
```

3.8 in Place reverse of word

- Very tough for me, please be careful

```
private static char[] reverseInPlace(string input)
    {
        char[] inputArray = input.ToCharArray();
        int len = input.Length;
        int start = 0, end = 0;

        while (end < len)
        {
            if (end != ' ')
            {
                while (end < len && inputArray[end] != ' ')
                    end++;
                reverseString(inputArray, start, end - 1);
                end++;
                start = end;
            }
        }
        return inputArray;
    }
    private static char[] reverseString(char[] inputArray, int start, int end)
    {
        while (end > start)
        {
            char temp=inputArray[end];
            inputArray[end] = inputArray[start];
            inputArray[start] = temp;
            end--; start++;
        }
        return inputArray;
    }
```

3.9 find frequency of any given word in a book

4 Stack and Queue

4.1 Implement Stack by using Array

```
public class Stack<T>
  {
    #region Properties
    private int _capacity;
    public int Capacity
```

```
{
   get
   {
      return _capacity;
   }
   set
   {
      _capacity = value;
   }
}
public int Length
{
   get
   {
      return Index + 1;
   }
}
private T[] _elements;
protected T[] Elements
{
   get
   {
      return _elements;
   }
   set
   {
      _elements = value;
   }
}
private int _index=-1;
public int Index
{
   get
   {
      return _index;
   }
   set
   {
      _index = value;
   }
}
#endregion
public Stack()
{
   Elements = new T[Capacity];
}
public Stack(int capacity)
{
   Capacity = capacity;
   Elements = new T[Capacity];
}

public void Push(T element)
{
   if (this.Length == Capacity)
   {
      IncreaseCapacity();
   }
```

```
            Index++;
            Elements[Index] = element;
        }
        public T Pop()
        {
            if (this.Length < 1)
            {
                throw new InvalidOperationException("Stack is Empty");
            }
            T element = Elements[Index];
            Elements[Index] = default(T);
            Index--;
            return element;
        }
        public T Peek()
        {
            if (this.Length < 1)
                throw new InvalidOperationException("Stack is Empty");
            return Elements[Index];
        }
        private void IncreaseCapacity()
        {
            Capacity++;
            Capacity *= 2;
            T[] newElements = new T[Capacity];
            Array.Copy(Elements, newElements, Elements.Length);
            Elements = newElements;
        }
    }
```

4.2

5 Linked List

5.1 remove duplicate from unsorted LinkedList

Crack 2.1

Special Problem

9.1 all subset of a set

Check in the cluster computer

9.2 all permutation of a string

Crack 8.4

www.mmrkhan.webs.com
khan4019@gmail.com

9.3 all valid combination of n-pair parenthesis

Crack 8.5

9.4 validate a combination of parenthesis (Sarthok Amazon)

Do yourself