Question 9)

ایده‌ی حل: فرض می‌کنیم آرایه‌ی اعداد به ۲ بخش تقسیم شده و تعداد جفت‌های نامرتب نیمه‌ی چپ و نیمه‌ی راست را به دست آورده‌ایم. واضح است که تعداد جفت‌های نامرتب کل آرایه = جفت‌های نامرتب سمت چپ + جفت‌های نامرتب سمت راست + جفت‌های نامرتبی که عضو اول (بزرگتر) آن‌ها در نیمه‌ی چپ و عضو دوم (کوچکتر) آن‌ها در نیمه‌ی راست آرایه است. در بخش ترکیب این مسئله از Merge sort کمک می‌گیریم.

```
int Reversepairs ( int a[], index l, index R)
{   if ( R <= l )
        return 0
    else if ( R - l == 1)
        if( a[l] > a[R])
            Swap ( a[l], a[R])
            return 1
        else
            return 0
    else
        index m = (R+l)/2

        int RpL = Reversepairs ( a, l, m)
        int RpR = Reversepairs ( a, m+1, R)
        int RpC = merge ( a, l, m, R)

        return (RpL + RpR + RpC)
}
```

```
int Merge( int a[], index l, index m, index R)
{   index k=0 , int num=0
    index i=l , j=m+1 ; int* S=New int [R-l+1]
    while( i≤m  and  j≤R)

        if( a[i] ≤ a[j])
            S[k]=a[i]
        else
            S[k]=a[j]
            num += m-i+1
        k++

    for( i ; i≤m ; i++)
        S[k]=a[i]
        k++
    for( j ; j≤R ; j++)
        S[k]=a[j]
        k++
    Copy S in a from index l to R

    return num
}
```

اگر آرایه مرتب باشد پس در صورتی که در مرحله‌ی merge کردن عنصری از سمت راست پیدا شود که از عنصر نام زیرآرایه‌ی سمت چپ کمتر باشد پس از عنصر i+1 ام و 2+i ام و ... m ام نیز کمتر است. این هم تعداد m-i+1 جفت نامرتب وجود دارد که عنصر مذکور در سمت راست با عنصرهای i تا m ام سمت چپ این جفت‌ها را می‌سازد.

Question 2)

$$F(n,k) = \sum_{1 \le i \le m \ \& \ i \le n} F(n-i, k-1) \longrightarrow \text{رابطه‌ی بازگشتی}$$

$$\begin{cases} F(0,k) = 0 & k \ge 0 \\ F(n,0) \to 1 & n \le m \\ F(n,0) \to 0 & n > m \end{cases} \longrightarrow \text{شرایط مرزی}$$

```
int F(int n, int k, int m)
{
    int a[n][k];
    for index i=0 to n
        if(i<=m)
            a[i][0] = 1
        else
            a[i][0] = 0
    for index j=0 to k
        a[0][j] = 0
    for index i=1 to n
        for index j=1 to k
            a[i][j] = 0
        for index l=1 to m
            if(l<=i)
                a[i][j] += a[i-l][j-1]
    return a[n][k]
}
```

شرایط مرزی

basic operation

Space $\in \theta(nk)$ — Complexity

Time Complexity $\in \theta(nkm)$

$i =$	1	2	3	...	n
تعداد basic operation	km	km	km		km

$\Rightarrow T(n) \in nkm \Rightarrow T(n) \in \theta(nkm)$

Arman

Question 3)

$F(n) = \max_{1 \le i \le n} \{F(n-i) + a_i\}$ → رابطه‌ی بازگشتی

$F(0) = 0 \quad F(1) = a_1$ → شرایط مرزی

```
int F( int n , int a[])
{
    int tab[n] ;                          → space ∈ Θ(n)
                                            Complexity
    tab[0] = 0   tab[1] = a[1]
    for index i = 2 to n
        int max = -∞                      → time ∈ Θ(n²)
                                            Complexity
        for index j = 1 to i
            k ← tab[i-j] + a[j]
            if (max < k)   // basic operation
                max = k
        tab[i] = max
    return tab[n]
}
```

i =	2	3	4 ...	n
تعداد تکرار basic operation =	2	3	4 ...	n

$\Rightarrow T(n) = \frac{n(n+1)}{2} - 1 \in \Theta(n^2)$

```
void printSolution(int n, int a[], int tab[])
{
    if (n >= 1)
    {   for index i = 1 to n

            if( tab[n-i] + a[i] == tab[n])
                Cout << i
                return printSolution(n-i, a, tab)
    }
}
```

تابع F بیشترین سود ممکنه را محاسبه می‌کند و پس تابع print solution چگونگی فروش برای رسیدن به این سود را نمایش می‌دهد.