Elif Gorgi
171044003

HOMEWORK 2

Part 1

1.

```
somefunction (rows, cols)
{
1   for (i=1 ; i<= rows ; i++)
    {
2       for(j=1 ; j<= cols ; j++)
3           print(*)
4       print(newline)
    }
}
```

	steps/exec	freq	total
1	2	rows + 1	2rows +2
2	2	rows.(cols +1)	2rows.cols + 2rows
3	1	rows. cols	rows. cols
4	1	rows	rows
		+	
			3rows.cols + 5rows +2

$T(rows,cols) = 3rows.cols + 5rows + 2$

$T_{worst}(rows, cols) = O(rows.cols)$

$T_{best}(rows, cols) = \Omega(rows.cols)$

$> T_{worst} = T_{best} \rightarrow T(rows,cols) = \Theta(rows.cols)$

* If best and worst cases equals each other, average case is equal to them either. So, average case is $\Theta(rows.cols)$

Elif Goral
171044003

2

```
somefunction (a,b)
{
1-   if (b==0)
2-      return 1

3-   answer = a
4-   increment = a

5-   for(i=1 ; i<b ; i++)   } b times
     {
6-      for(j=1 ; j<a ; j++)   } (b-1) a times
        {
7-         answer += increment
        }
8-      increment = answer
     }
9-   return answer
}
```

	steps/exec	freq	total
1-	1	1	1
2-	1	1	1
3-	1	1	1
4-	1	1	1
5-	2	b	2b
6-	2	(b-1) a	2ab - 2a
7-	1	(b-1) (a-1)	ab-a-b+1
8-	1	b-1	b-1
9-	1	1	1
		+	
			3ab - 3a + 2b + 5

$T_{worst}(a,b) = O(a.b)$

$T_{best}(a,b) = \Omega(1)$

Elif Goral
171044003

3

```
somefunction ( arr[], arr_len)
{
1-    val = 0
2-    for (i = 0; i< arr_len /2 ; i++)      } arr_len/2 +1 times
3-        val = val + arr[i]

4-    for (i = arr_len/2 ; i< arr_len ; i++)    } arr_len/2 +1 times
5-        val = val - arr[i]

6-    if( val >=0)
7-        return 1
8-    else
9-        return -1
}
```

	steps1exec	freq	total
1	1	1	1
2	2	arr_len /2 +1	arr_len +2
3	2	arr_len /2	arr_len
4	2	arr_len/2 +1	arr_len +2
5	2	arr_len/2	arr_len
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
		+	
			4 arr_len +9

$T_{worst}(arr\_len) = O(arr\_len)$
$T_{best}(arr\_len) = \Omega(arr\_len)$ $\Rightarrow$ $\Theta(arr\_len) = T(arr\_len)$

Elif Gool
171044003

4

```
somefunction(n)
{
1-   c=0
2-   for(i=1 to n*n)
3-     for(j=1 to n)
4-       for(k=1 to 2*j)
5-         c= c+1
6-   return c
     }
```

	steps/exec	freq	total
1	1	1	1
2	2	n^2+1	$2n^2+2$
3	2	$n^2(n+1)$	$2n^3+2n^2$
4	2	$n^2[(n+1)^2-1]$	$2n^4+4n^3$
5	2	$n^2 . n(n+1)$	$2n^4+2n^3$
6	1	1	1
		+	
			$4.n^4+8n^3+4n^2+4$

$1+3+5+\dots+2k-1 = \sum 2k+1 = k^2$

$3+5+\dots+2k-1 = k^2-1$

for()
  for()
    for() } k^2-1 } $\xrightarrow{k=n+1}$ $n^2[(n+1)^2-1]$

$1+2+3+\dots+2k = \sum 2k = \frac{k(k+1)}{2}$

$2+4+6+\dots+2n = 2(1+2+\dots n) = \frac{2n(n+1)}{2}$

for()
  for()
    for() } $n(n+1)$ } $n^2(n+1).n$

$T_{best} = \Omega(n^4)$
$T_{worst} = O(n^4)$ $\Rightarrow$ $T(n) = \Theta(n^4)$

Elif Gonul
171044003

5

```
otherfunction (xp, yp)
{
  temp = xp;
   xp = yp
   yp = temp
}

somefunction (arr[], arr_len)
   for(i=0; i<arr_len-1; i++)   } arr_len -1 times
   {
      min_idx = i
      {for (j=i+1 ; j < arr_len ; j++)
          if (arr[j] < arr [min_idx])
             min_idx = j
        otherfunction (arr[min_idx], arr[i])
      }
   }
```

(arr_len-1) + (arr_len-2) + ... (arr_len - arr_len)

$$\frac{(arr\_len-1)(arr\_len)}{2}$$

$$T(arr\_len) = (arr\_len-1)\frac{(arr\_len)(arr\_len-1)}{2} = \frac{arr\_len^3 - 2\,arr\_len^2 + arr\_len}{2}$$

$$O(arr\_len) = arr\_len^3$$

$T_{worst}(arr\_len) = O(arr\_len^3)$
$T_{best}(arr\_len) = \Omega(arr\_len^3)$

$$\Rightarrow \boxed{\Theta(arr\_len^3) = T(arr\_len)}$$

Elif Gonul
171044003

6.

```
otherfunction (a,b)
{
  if(b==0)
    return 1

  answer = a
  increment = a
  for(i=1 to b:          } b times
  {
    for j=1 to a         } a times
      answer += increment
      increment = answer
  }
  return answer
}

somefunction (arr,arr_len)
{
  for i=0 to arr_len;
    for j=i to arr_len;
      if (otherfunction (n % i,2) == arr[i])
        print (arr[i])
  }
```

otherfunction → $T_1(a,b)_{worst} = O(a.b)$

$T_1(a,b)_{best} = \Omega(1)$

somefunction → $\leq = 1+2+3+\dots+arr\_len+arr\_len+1$

$(arr\_len).\frac{(arr\_len+1)(arr\_len+2)}{2}$

$T_2(arr\_len) = \frac{arr\_len^3 + 3\,arr\_len^2 + 2\,arr\_len}{2} \cdot O(a,b)$

$T_2(arr\_len)_{worst} = O(arr\_len^3).O(2a) = O(a, arr\_len^3)$

$T_{best} = T_{worst} = O(a.arr\_len^3) = \boxed{\Theta(a.arr\_len^3)}$

$T_{best} = O(a_{arr\_len^3})$

Elif Gorol
171041003

7

```
otherfunction (x,i)
{
   s=0
   for(j=1 ; j<=i ; j=j*2)
       s = s + x[j]
   return s
}

somefunction (arr_len, arr[])
{
   for(i=0; i<=arr_len-1; i++)   { arr_len+1 times }
       A[i] = otherfunction(arr,i)/(i+1)
   return A
}
```

otherfunction → $T(i) = O(\log_2 i)$

$1.2 = 2$
$2.2 = 2^2$
$2^2.2 = 2^3$
$\vdots$

$2^k = i \rightarrow \boxed{\log_2 i = k}$

somefunction → $T(arr\_len) = O(arr\_len \cdot \log_2(arr\_len-1))$

$T_{worst} = O(arr\_len \cdot \log_2(arr\_len-1))$

$T_{best} = \Omega(arr\_len \cdot \log_2(arr\_len-1))$

$T_{worst} = T_{best}$

↳ $\boxed{T(arr\_len) = \Theta(arr\_len \cdot \log_2(arr\_len-1))}$

Dij Gorol
17104403

8.

```
somefunction (n)
{
    res = 0
    j = 1
    if(n < 10)
       return  n+10

    for(i=9 ; i>=1 ; i--)        ; 10 times → constant
       while (n %i ==0)
            n = n/i
            res = res+ j *i
            j *= 10
    if(n>10)
       return -1
    return res
}
```

$T_{best} = \Omega(1)$

Elif Gorol
171044003

Part 2

1

```
func1(String array[][], int arr_i, int arr_j, int given_i, int given_j)
{
 distance;
 min = 100
 arr[]
 for(i=0; i<arr_i; i++){          ) arr_i times
    for(j=0; j<arr_j; j++){       ) arr_j times
       if(array[i][j] != ""){
          distance = sqrt(pow(i-given_i,2) + pow(j-given_j,2))
          if(distance < min){
             min = distance
             arr[0] = i
             arr[1] = j
          }
       }
    }}
    return arr
}
```

$T(arr\_i, arr\_j) = O(arr\_i * arr\_j)$

$T_{worst} = O(arr\_i * arr\_j)$

$T_{best} = \Omega(arr\_i * arr\_j)$ $\Rightarrow T(arr\_i, arr\_j) = \Theta(arr\_i * arr\_j)$

→ while finding the closest point in the array to a given point, I use hypotenuse theorem. But those processes in become constant time. For this reason only loop's repetitions are effect the complexity.

Elif Gorol
171044023

2

a)
```
func2_a (int A[], int arr_len){
    for(i=1; i<arr_len-1; i++){   } arr_len-1 times
        if( A[i] <= A[i+1] && A[i] <= A[i-1]){
            return A[i]
        }
    }
    return -1
}
```

$T(arr\_len) = O(arr\_len)$
$T(arr\_len) = arr\_len - 1$

$T_{worst} = O(arr\_len)$
$T_{best} = \Omega(1)$

→ If "i" fulfill the codition, loop return only one time.
So, best case is 1.

---

b)
```
func2_b (int A[], int or_len){
    int x[or_len].
    k=0
    for(i=1; i<or_len-1; i++)   } or_len-1 times
        if(A[i] <= A[i+1] && A[i] <= A[i-1])
            x[k] = A[i]
            k++
    }
    return x
}
```

* function must return the loop completely to find all local minimum points.

$T(or\_len) = or\_len - 1$
$T(or\_len) = O(or\_len)$
$T_{best} = \Omega(or\_len)$
$T_{worst} = O(or\_len)$ → $\boxed{T(or\_len) = \Theta(or\_len)}$

Elif Gonul
171041003

3

```
func3 (int array[], int number, int arr_len)
{
    for (int i=0; i< arr_len; i++)        } arrlen+1 times
    {
        if(array[i] * 2 == number)
            return true
        for (int j=i+1 ; j< arr_len;j++){   } arr_len.(arr_len+1)/2 times
            if(array[i] + array[j] == number)
                return true
        }
    }
    return false
}
```

when j=1 → 4 times
j=2 → 3 times
j=3 → 2 times
j=4 → 1 time
$\frac{n.(n+1)}{2}$

$$T(arr\_len) = (arrlen+1)\ \frac{(arr\_len)\ (arr\_len+1)}{2}$$

$$T(arr\_len) = \frac{arr\_len^3 + 2arr\_len^2 + arr\_len}{2}$$

$$T_{worst} = O(arr\_len^3)$$

$$T_{best} = \Omega(arr\_len)$$

* Function checks at first whether it's the sum of the some two numbers. Then it looks at the other elements of one.

Elif Gonof
171044003

4

```
func4 (int array[] , int arr_len)
{
   for (i=1 ; i<arr_len ;i++){ } arrlen times
       if (func3 (array, array[i], arr_len)==false)
        return false
    }
    return true
}
```

$T(arr\_len) = O(arr\_len)$

$T_{worst} = O(arr\_len)$
$T_{best} = \Omega(1)$

* That function checks the array is sum chain of length or not. with using function 3.