FATİH KAAN SALGIR - 171044009

PART 1

Question 1

```
somefunction (rows, cols) {                steps   freq      total
    for (i=1; i <= rows ; i++) {             2     r+1       2r+2
        for (j=1 <= cols ; j++)              2     r+(c+1)   2r+2c+2
            print (*)                        1     r*c       r*c
        print (newline)                      1     r         r
    }                                                    +
}                                                        5r+rc+4
```

Loops will be iterated till 'rows' and 'cols', regardless of body. Inner for loop takes linear time, proportional to c, and outer loop will also takes linear time, proportional to n. Therefore complexity is $\Theta(r * c)$.

Question 2

```
somefunction (a, b) {                      steps   freq      total
    if (b == 0)                              1       1         1
        return 1                             1       1         1
    answer = a                               1       1         1
    increment = a                            1       1         1
    for (i=1; i < b; i++) {                  2       b         2b
        for (j=1; j< a; j++) {               2       (b-1)a    2ba-2a
            answer += increment              2       (a-1)(b-1) 2ab-2a-2b+2
        }
        increment = answer                   1       b-1       b-1
    }
    return answer                            1       1         1
}                                                              +
                                                               4ab-4a+6
```

It might enter if statement, it takes constant time;

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

or, it could enter for loops, in this case;

$T_{worst} = \Theta(a \times b)$

Therefore in total we have $O(a \times b)$.

Question 3

	steps	freq	total
`somefunction(arr[], arr_len) {`			
`val = 0`	1	1	1
`for (i=0; i < arr_len / 2; i++)`	2	$(n/2)+1$	$n+2$
`val = val + arr[i]`	2	$n/2$	n
`for (i= arr_len /2; arr_len; i++)`	2	$(n/2)+1$	$n+2$
`val = val - arr[i]`	2	$n/2$	n
`if (val >= 0)`	1	1	1
`return 1`	1	1	1
`else`			
`return -1`	1	1	+ 1
`}`			$4n+8$

If we ignore base cases, complexity of loops are $\Theta(\frac{n}{2})$. 'if' statements takes constent time.

$\Theta(\frac{n}{2} + \frac{n}{2} + 1)$

constants are insignificant;

$\Theta(n)$.

Question 4

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

Very inner loop's constrain is determined by 'j', which incremented linearly. We can observe behaviours of these 2 loops together.

$A = 2 + 4 + 6 + ... + 2n$

$= 2(1+2+3+...+n)$

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

$A = (n)(n+1) = n^2+n$.

Outer loop has complexity of $\Theta(n^2)$

$\Theta(n^2 \times (n^2+n))$

$= \Theta(n^4)$

Question 5

'otherfunction()' consist of basic assignments. It will take constant time. We can say it is $\Theta(1)$.

To analyze 'somefunction()', we will start with the very inner loop. In the loop there is on if statement. Condition part of the 'if' is comparasion operation, it will take constent time. Body of the if statement also takes constant time, since it is assignment.

For loop will iterate increasingly every time deperding on i. If we sum all incrementations, we obtain;

$$\frac{n \times (n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$$

Since low order elements and constents are insignificant, complexity of this loop is $\Theta(n)$.

'min_idx = i.' will take constant time.

Outer loop will iterate until arr_len-1, regardless of its body. Therefore it is $\Theta(n)$.

In total we have; $\Theta(n \times n) = \Theta(n^2)$

Question 6

'otherfunction()' has the time complexity $\theta(a \times b)$ by itself, since it consist of 2 for loops. But otherfunction will be called by 2. Therefore we can consider it as linear time.

'somefunction()' has behaviour of summation of consecutive numbers;

$$\frac{n \times (n+1)}{2} \rightarrow \theta(n^2).$$

In if statement otherfunction will be called some 'a' parameter. Therefor in total we have;

$$\theta(a \times n^2).$$

Question 7

'otherfunction()' has for loop matters in terms of complexity. j multiplied by a constant, thus it has logarithmic behaviour.
'somefunction()' will iterate until 'arr_len'. It will call otherfunction with parameter i. To approximate; if it be called by n, it would be log(n).

i	otherfunction()
1	log(1)
2	log(2)
⋮	⋮
N	log(N)

Outer loop will iterate till N, so the complexity must be greater than $\Theta(n)$. To conclude complexity is either

$O(n \times \log(n))$ or $\Omega(n)$.

Question 8

Assignments take constant time.

Condition of if statement is comparison, it takes constant time. Body of the if statement also takes constant time, since it is a return times. We can omit it since it won't affect time much.

Run time of the while loop depends on both i and n.
first if → $\Theta(1)$
while → $T_{worst}: \Theta(\log(n))$, $T_{best}: \Theta(1) \rightarrow O(\log(n))$
for → 9 times, $\Theta(1)$
second if → $\Theta(1)$

result: $O(\log(n))$

PART 2

Question 1

```
calculate_distance(point1, point2) {
    x = point2.x - point1.x
    y = point2.y - point1.y
    return sqrt(x^2 + y^2)
}
find(arr, given_point) {
    closest = calculate_distance(arr[0], given_point)
    for (i=1 ; i < arr_len ; i++) {
        x = calculate_distance(arr[i], given_point)
        if( x < closest)
            closest = x
    }
    return closest
}
```

calculate_distance: $\Theta(1)$

find: $\Theta(1) \times \Theta(1) \times \Theta(n) = \Theta(n)$

Question 2

```
find_local_mins(arr, arr_len) {
    local_mins[]
    j=0
    for (i=1 ; i < arr_len ; i++)
        if (arr[i] >= arr[i+1] && arr[i] <= arr[i+1]) {
            local_mins[j] = arr[i]
            j++
        }
    }
    return local_mins
}
```

Assignments take constant time. Loop will be iterate regardless of if statement, 'arr_len' times.

Complexity: $\Theta(n)$

Question 3

```
has_sum(arr, arr_len) {
    check = arr[arr_len-1]
    arr_len -= 1
    for(i=0; i<arr_len; i++)
        for(j=i; j<arr_len; j++)
            if(arr[i] + arr[j] == check)
                return true
    return false
}
```

} $\Theta(1)$

$T_{best} = \Theta(1)$ $T_{worst} = \Theta(n^2)$

result: $O(n^2)$ (if iterates through whole array).

Question 4

```
is_sum_chain(arr, arr_len) {
    check_size = 2
    for(i=check_size; i<arr_len; i++) {
        if(has_sum(arr, i) == false)
            return false
    return true
}
```

→ $O(n)$

→ $O(n^2)$

$O(n \times n^2) = O(n^3)$.