1) Find the **running time** \( T(n) \) and the asymptotic running time (using \( \Theta \)-notation and \( O \)-notation) of the following piece of code:

```python
n=eval(input("Enter an integer number greater than 2:"))
for i in range(n):
    print(i)  # 1 step
for j in range(n):
    print(j)  # 1 step
```

\[ T(n) = n+n = 2n \]
\[ T(n) = \Theta(n) \]

2) Find the **running time** \( T(n) \) and the asymptotic running time (using \( \Theta \)-notation and \( O \)-notation) of the following piece of code:

```python
n=eval(input("Enter an integer number greater than 10:"))
for i in range(n):
    for j in range(n):
        print(i,"\t",j)  # 1 step
```

\[ T(n) = n*n*1 = n^2 \]
\[ T(n) = \Theta(n^2) \]

3) Find the **running time** \( T(n) \) and the asymptotic running time (using \( \Theta \)-notation and \( O \)-notation) of the following piece of code:

```python
n=eval(input("Enter an integer number greater than 12:"))
while n>1:
    print(n)  # 1 step
    n = n//2  # each time n becomes “twice less”, until eventually it is <= 1, 2 steps
print(n)  # 1 step
```

\[ T(n) = 3\log_2 n + 1 \]
\[ T(n) = \Theta(\log n) \]
1) Copy the following program (you may omit the docstring):

```python
def summation1(n):
    """ finds the sum (n+i)^2/i, where i runs from 1 to n
    """
    sum = 0
    for elem in list(range(n)):
        sum += (n+1+elem)**2/(elem+1)
    return sum
```

2) Run the defined procedure on different inputs, for example \( n = 1, 2, 10 \).
Write down the results.

\[
\begin{align*}
4.0 \\
17.0 \\
547.8968253968254
\end{align*}
\]

3) Write, following the code of the program, each call of this procedure on inputs \( n = 1, 2, 10 \) as a sum of fractions, i.e. write which sum finds for procedure for each of these calls, but don't calculate it.

\[
\begin{align*}
\text{n=1} & \quad \text{n=2} & \quad \text{n=10} \\
\text{list: 0} & \quad \text{list = 0,1} & \quad \text{list = 0,1,2,3,4,5,6,7,8,9} \\
\text{sum: } 0 + \frac{2^2}{1} & \quad \text{sum: } 0 + \frac{3^2 + 4^2}{2} & \quad \text{sum: } 0 + \frac{11^2}{1} + \frac{12^2}{2} + \frac{13^2}{3} + \frac{14^2}{4} + \frac{15^2}{5} + \frac{16^2}{6} + \frac{17^2}{7} + \frac{18^2}{8} + \frac{19^2}{9} + \frac{20^2}{10} \\
\text{as expected} & \quad \text{as expected} & \quad \text{as expected}
\end{align*}
\]

4) Find the running time of the procedure (depending on \( n \)), assuming that it takes one unit of time for each of math operations; the assignment operator and `range` function take also one time unit, and function `list` takes \( n \) time units.

\[
\begin{align*}
\text{sum = 0} & \quad \text{1 step/ time unit} \\
\text{for elem in list(range(n)):} & \quad \text{range(n): 1 step; list: n steps} \\
\text{n iterations} & \quad \text{7 steps} \\
\text{sum += (n+1+elem)**2/(elem+1)} & \quad \text{7n steps} \\
\text{return sum} & \quad \text{1 step}
\end{align*}
\]

Therefore, \( T(n) = 1+1+n+7n+1 = 3+8n \)

5) What is the order of growth (in terms of \( O \) and \( \Theta \))? 

\( O(n), \Theta(n) \)
1) Copy the following program (you may omit the docstring):

```python
def summation2(n):
    """ finds the sum 2^i/i, where i runs from 1 to n
    pre: n in positive integer
    post: returns a positive integer number."""
    sum = 0
    for elem in list(range(n)):
        sum += 2**(elem+1)/(elem+1)
    return sum
```

2) run the defined procedure on different inputs, for example \( n = 1, 2, 10 \).
Write down the results.

```
2.0
4.0
237.30793650793652
```

3) Write, following the code of the program, each call of this procedure on inputs \( n = 1, 2, 10 \) as a sum of fractions, i.e. write which sum finds for procedure for each of these calls, but don't calculate it.

```
n=1  n=2  n=10
list: 0 list = 0,1 list = 0,1,2,3,4,5,6,7,8,9
sum: 0+ 2^1 2^1 2^1/1 2/1 2^2/2
as expected as expected as expected
```

4) find the running time of the procedure (depending on \( n \)), assuming that it takes one unit of time for each of math operations; the assignment operator and `range` function take also one time unit, and function `list` takes \( n \) time units.

```
sum = 0 1 step/ time unit
for elem in list(range(n)): range(n): 1 step; list: n steps
n iterations
    sum += 2**(elem+1)/(elem+1)
    6 steps 6n steps
return sum 1 step
```

Therefore, \( T(n) = 1+1+n+6n+1 = 3+7n \)

5) What is the order of growth (in terms of \( O \) and \( \Theta \))? 

\( O(n), \Theta(n) \)
1) Copy the following program (you may omit the docstring):

```python
def summation3(n):
    """ finds the sum i^2/(i+1), where i runs from 1 to n
    pre: n in positive integer
    post: returns a positive integer number."""
    sum = 0
    for elem in list(range(n)):
        sum += (elem+1)**2/(elem+2)
    return sum
```

2) run the defined procedure on different inputs, for example $n = 1, 2, 10$.
Write down the results.

0.5
1.8333333333333333
47.019877344877344

3) Write, following the code of the program, each call of this procedure on inputs $n = 1, 2, 10$ as a sum of fractions, i.e. write which sum finds for procedure for each of these calls, but don't calculate it.

<table>
<thead>
<tr>
<th>$n=1$</th>
<th>$n=2$</th>
<th>$n=10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>list: 0</td>
<td>list: 0,1</td>
<td>list: 0,1,2,3,4,5,6,7,8,9</td>
</tr>
<tr>
<td>sum: $0+\frac{1^2}{2}$</td>
<td>sum: $0+\frac{1^2}{2}+\frac{2^2}{3}$</td>
<td>sum: $0+\frac{1^2}{2}+\frac{2^2}{3}+\frac{3^2}{4}+\frac{4^2}{5}+\frac{5^2}{6}+\frac{6^2}{7}+\frac{7^2}{8}+\frac{8^2}{9}+\frac{9^2}{10}+\frac{10^2}{11}$</td>
</tr>
<tr>
<td>as expected</td>
<td>as expected</td>
<td>as expected</td>
</tr>
</tbody>
</table>

4) find the running time of the procedure (depending on $n$), assuming that it takes one unit of time for each of math operations; the assignment operator and `range` function take also one time unit, and function `list` takes $n$ time units.

```plaintext
sum = 0
for elem in list(range(n)):    range(n): 1 step;    list: n steps
n iterations
    sum += (elem+1)**2/(elem+2)
6 steps
return sum    1 step
```

Therefore, $T(n) = 1+1+n+6n+1 = 3+7n$

5) What is the order of growth (in terms of $O$ and $\Theta$)?

$O(n)$, $\Theta(n)$