1. **(30 marks)** For each of the algorithms below, give a tight asymptotic worst-case, \(i.e.,\) Big-Oh, time complexity functions \(O(f(n))\). Briefly explain the reasoning behind each derivation.

(a) **(5 marks)**

```python
sum = 1
for i = 1 to n do
    sum = sum / i
    k = i * i
    for j = 1 to n do
        sum = sum * (k - 13)
    for j = 1 to n do
        sum = sum * (k - 13)
    sum = sum * 14 * 1000000
for i = 1 to n * n do
    sum = sum + i
sum = sum - 42
```

(b) **(5 marks)**

```python
sum = 13
cond = false
for i = 1 to n do
    for j = 1 to 7 do
        sum = sum / (i + j)
    if COND(sum)
        cond = true
    else
        for j = 1 to n do
            sum = sum - j
    if cond
        for j = 1 to n * log(n) do
            sum = sum + (i/j)
```

Note that method \(COND()\) runs in 5 timesteps and method \(\log(n)\) returns the logarithm (base 2) of \(n\), \(i.e.,\) \(\log_2 n\).
(c) (5 marks)

sum = 157
cond = false
for i = 1 to 7 do
  for j = 1 to n do
    sum = sum * (i/j)
    if (sum > 23)
      sum = sum + 23
      k = sum
      sum = sum + k
    else
      k = sum - 23
      for k = 1 to log(n) * n do
        sum = sum - (k/j)
  if (cond)
    sum = sum - 256

(d) (5 marks)

sum = 42
for i = 1 to n * n do
  j = 1
  finished = true
  while ((j <= n) and (not finished)) do
    for k = 1 to log(n) * n do
      sum = sum / (k * i) + j
    if COND(sum)
      finished = true

Note that method COND() runs in \((n + 13)\) timesteps.

(e) (5 marks)

sum = 42
for i = 1 to n * n do
  j = 1
  finished = false
  while ((i <= n) and (not finished)) do
    for k = 1 to log(n) * n do
      finished = true
    if COND(sum)
      sum = sum / (k * i) + j

Note that method COND() runs in \((n + 13)\) timesteps.
(f) (5 marks)

\[
\text{sum} = 42 \\
\text{for } i = 1 \text{ to } n \log(n) \text{ do} \\
\quad j = 1 \\
\quad \text{finished} = \text{false} \\
\quad \text{for } k = 1 \text{ to } n \text{ do} \\
\quad \quad \text{if } \text{COND(sum)} \\
\quad \quad \quad \text{sum} = \text{sum} / (k \times i) + j \\
\quad \quad \text{while } ((j \leq n) \text{ and } \text{(not finished)}) \text{ do} \\
\quad \quad \quad \text{finished} = \text{true}
\]

Note that method \text{COND()} runs in \((n + 13)\) timesteps.

2. (30 marks) Prove or disprove the following:

(a) (10 marks) \( f(n) = (n - 2)(n - 6) \) is not \( \Theta(n^2) \).

(b) (10 marks) \( f(n) = n^d + 10n^2 \), where \( d \) is some integer constant greater than or equal to 2, is \( O(n^d) \).

(c) (10 marks) \( f(n) = 10^{127}2^n \) is \( \Omega(3^n) \).

All proofs should give appropriate bounds on values of \( c \) and \( n_0 \).

3. (24 marks) Solve the following recurrences using the iteration method:

(a) (12 marks)

\[
T(n) = \begin{cases} 
27 & n \leq 4 \\
T(n - 4) + cn^2 & n > 1 
\end{cases}
\]

(b) (12 marks)

\[
T(n) = \begin{cases} 
3 & n \leq 1 \\
8T(n/2) + cn & n > 1 
\end{cases}
\]
4. **(16 marks)** For each of the recursive algorithms below, derive a recurrence describing the exact running time of that algorithm.

(a) **(8 marks)**

```
FUNKY-REC1(n, x)
    if (n <= 2)
        print x
    else
        for i = 1 to n * log(n) do
            if i is odd then
                x = x + i
            else
                x = x - 1
        FUNKY-REC1(n - 2, x)
        x = x * x
        FUNKY-REC1(n - 3, x - 1)
        for i = 1 to n do
            x = x / i
        FUNKY-REC1(n - 2, x)
```