Recurrences and Asymptotic Complexity

1. Solve each recurrence, using O, Ω, or Θ, whichever is appropriate.

   (a) $F(n) = 4F\left(\frac{n}{2}\right) + 5n^2$

   (b) $f(n) = f(n-1) + n.$

   (c) $f(n) = f\left(\frac{n}{2}\right) + f\left(\frac{n}{3}\right) + n$

   (d) $f(n) = f(\sqrt{n}) + 1.$

   (e) $f(n) = 2f(\sqrt{n}) + \log n$

   (f) $H(n) = 2H\left(\frac{n}{2}\right) + O(n)$

   (g) $g(n) = 2g(n-1) + 1$

   (h) $G(n) \geq G(n-1) + \lg n$

   (i) $H(n) \leq 2H(\sqrt{n}) + 4.$

   (j) $K(n) = K(n - 2\sqrt{n} + 1) + n.$

   (k) $F(n) \leq F\left(\frac{n}{5}\right) + F\left(\frac{7n}{10}\right) + n$

   (l) $F(n) = 2F\left(\frac{2n}{3}\right) + F\left(\frac{n}{3}\right) + n$

   (m) $f(n) = 1 + f(\log n)$

2. Write the asymptotic time complexity for each code fragment, giving the answer in terms of n, using O, Ω, or Θ, whichever is appropriate.

   (a)
```
for (int i=1; i < n; i++)
  for (int j=i; j > 0; j--)
    cout << "hello world" << endl;
```

   (b)
```
for (int i=1; i < n; i++)
  for (int j=1; j < i; j++)
    cout << "hello world" << endl;
```

   (c)
```
for (int i=1; i < n; i = 2*i)
  for (int j=1; j < i; j++)
    cout << "hello world" << endl;
```

   (d)
```
for (int i=1; i < n; i++)
  for (int j=1; j < i; j = j*2)
    cout << "hello world" << endl;
```

   (e)
```
for (int i=1; i < n; i++)
  for (int j=i; j < n; j = j*2)
    cout << "hello world" << endl;
```

   (f)
```
for (int i=2; i < n; i = i*i)
  cout << "hello world" << endl;
```

   (g)
```
for (int i=1; i*i < n; i++)
  cout << "hello world" << endl;
```

   (h)
```
for (int i=n; i > 1; i = i/2)
  for (int j=1; j < i; j=2*j)
    cout << "hello world" << endl;
```

   (i) For this problem, `george` is a function which returns an integer. You have no idea what that integer will be.

```
int m = n;
while(m > 0){
  int g = george(m);
  if (g > 0) m = m - g;
  else m = m - 1;
  cout << "hello world" << endl;
  }
```