Establish the required upper bound for the worst-case time complexity in questions 1 and 2. You need to provide and justify a big-O bound. Your bound should be tight in order to receive full credit, but you do not need to prove that it is tight.

1. [10 marks] Explain why the following pseudocode terminates. Then, give an upper bound for the time complexity as a function of $n$. You can assume that each evaluation of condition_1 and condition_2 takes constant time, and does not change the values of $i$, $j$, and $n$. However, the truth values of the conditions may change during the execution of the pseudocode.

```plaintext
i := 1;
j := n; /* a positive integer */
while (i < j) do
  if (condition_1) then
    i := 2*i;
  else if (condition_2) then
    i := i+j;
j := 3*j-i ;
else
  i := i+2;
j := j+1;
```
2. [10 marks] Consider the following recursive algorithm studied in class for multiplying two integers. Establish a big-$O$ upper bound for the number of times the function calls itself, as a function of $b$.

```
function multiply(a,b)
if b = 0 then return 0
else if b is even then
    t := multiply(a, b/2);
    return t+t;
else if b is odd then
    t := multiply(a, b-1);
    return a+t;
```

3. [10 marks] For long integer multiplication (computing $a \times b$), the best algorithm currently known has time complexity $f(n) = n(\log n)^2\Theta(\lg^*(n))$. Here $n$ is the size of the input (i.e., the total number of bits in $a$ and $b$). (The function $\lg^*$ is defined in the textbook, Chapter 3.)

Joe claims that he has achieved two new results: (a) a proof that no algorithm can perform integer multiplication in $o(f(n))$ time and (b) an algorithm that takes $O(n \log n)$ time to compute $a^2$, where $n$ is the number of bits of $a$. Do you believe him? Justify your answer.