Written Assignment 1 Sample Answer

Problem 1:

(a) Answer:

```
int f1 (int n) {
    int res = 0; …………………1
    for(int i = 0; i < n; i++) …….3n+2
        res ++; ………………...2n
    return res; …………………..1
}
```

Total: 5n+4

(b) Answer:

```
int f2 (int n) {
    int res = 0; ……………………1
    for(int i = 1; i < n; i*=2) …........3log2(n) + 2
        for(int j = 0; j < n; j++) ….(3n + 2) × log2(n)
            res += i * j; …………3n log2(n)
    return res; …………………….1
}
```

Total: $6n\log_2(n) + 5\log_2(n) + 4$

(c) Answer:

```
int f3(int n) {
    if(n<=0)
        return 0;
    return 2 * f3(n-2) + 1;
}
```

If the basic operation of $f_3(n)$ is $T(n)$.
For base case, the code segment is:

```
if(n<=0)
    return 0;
```

It concludes 2 basic operations. That is, $T(n) = 2, n \leq 0$

For the general cases, if the basic operation of $f_3(n)$ is $T(n)$, then we have the recurrence relation as follows:

$T(n) = T(n-2) + 5, n > 0$, since:

(1) The judgement of "$n \leq 0$", concludes 1 basic operation;

(2) For arithmetic operations in the code "return 2 * f3(n-2) + 1",

a) the multiplication operation, the minus operation and plus operation in "$2 * f_3(n-2) + 1$", concludes 1 basic operation respectively, that is 3 basic operations. And "return" concludes 1 basic operation;

b) This statement "return 2 * f3(n-2) + 1" concludes 4 basic operations totally.

All in all,

$$T(n) = \begin{cases} T(n-2) + 5, & n > 0 \\ 2, & n \le 0 \end{cases}$$

$$\begin{aligned} T(n) &= T(n-2) + 5 \\ &= T(n-4) + 5 + 5 \\ &= T(n-4) + 10 \\ &= T(n-6) + 15 \\ &= \cdots \\ &= T(n - 2 * k) + 5 * k \end{aligned}$$

Let $k = \frac{n}{2}$, $T(n) = 5 * \frac{n}{2} + 2$. (Here, *n* is always a power of 2)

Total: $5 * \frac{n}{2} + 2, n > 0$

Problem2:

(a) Answer:

Conclusion: $f(n) = O\big(g(n)\big)$, $f(n) = \Omega\big(g(n)\big)$, $f(n) = \Theta\big(g(n)\big)$

$f(n) = O\big(g(n)\big)$ means there are positive constants *c* and n_0 such that $f(n) \le c\, g(n)$ when $n \ge n_0$. For large n, $2n$ dominates $\log_2 n$. Therefore, let's choose $c = 2$ and $n_0 = 1$ to satisfy $n \le 2 \times (2n + \log_2 n)$. So, $f(n) = O\big(g(n)\big)$ holds.

$n_0 = 1$ means there are positive constants *c* and n_0 such that $f(n) \ge c\, g(n)$ when n $\ge n_0$. Let's choose $c = \frac{1}{4}$, thus we have:

$$n \ge \frac{1}{4}(2n + \log_2 n) \Longrightarrow n \ge \frac{1}{2}\log_2 n \Longrightarrow n \ge 2.307$$

Thus, we choose $c = \frac{1}{4}$ and $n_0 = 3$, this inequity will hold. So, $f(n) = \Omega\big(g(n)\big)$ holds.

Because both $f(n) = O\big(g(n)\big)$ and $f(n) = \Omega\big(g(n)\big)$ hold, $f(n) = \Theta\big(g(n)\big)$ holds.

(b) Answer:

Conclusion: $f(n) = O\big(g(n)\big)$, $f(n) \ne \Omega\big(g(n)\big)$, $f(n) \ne \Theta\big(g(n)\big)$

$f(n) = O\big(g(n)\big)$ means there are positive constants *c* and n_0 such that $f(n) \le c\, g(n)$ when $n \ge n_0$. Therefore, choose $c = 1$ and $n_0 = 1$ to satisfy $n^2 \le n^3$. Therefore, $f(n) = O\big(g(n)\big)$ holds.

$f(n) = \Omega\big(g(n)\big)$ means there are positive constants *c* and n_0 such that $f(n) \ge c\, g(n)$ when n $\ge n_0$. This is not possible because $g(n) = n^3$ grows faster than $f(n) = n^2$.

$f(n) = \Theta\big(g(n)\big)$ holds when $f(n) = O\big(g(n)\big)$ and $f(n) = \Omega\big(g(n)\big)$. Since $f(n) \ne \Omega\big(g(n)\big)$, $f(n) \ne \Theta\big(g(n)\big)$.

Problem3:

(a) Answer:

If $f(n)=\frac{1}{n}$, $g(n)=f(n)+1=\frac{1}{n}+1$, and $\lim_{n\to\infty} f(n)=0$, and $\lim_{n\to\infty}(f(n)+1)=1$, thus, we cannot find such n_0 and c to make that when $n>n_0$, $g(n)\le cf(n)$. Thus, $g(n)\ne\Omega\big(f(n)\big)$, and $f(n)\ne\Theta(f(n)+1)$.

(b) Answer:

If $f(n)=n!$, then $f(n+1)=(n+1)!$.When $n\to\infty$, $\lim_{n\to\to\infty}\frac{f(n+1)}{f(n)}=\infty, so\ f(n)\ne\Theta(f(n+1))$.

Problem4 Answer:

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right)+n \\
&= T\left(\frac{n}{4}\right)+\frac{n}{2}+n \\
&= T\left(\frac{n}{8}\right)+\frac{n}{4}+\frac{n}{2}+n \\
&= \cdots \\
&= T\left(\frac{n}{2^i}\right)+n\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{i-1}}\right) \\
&= T\left(\frac{n}{2^i}\right)+n\left(\frac{1\times\left(1-(\frac{1}{2})^i\right)}{1-\frac{1}{2}}\right) \\
&= T\left(\frac{n}{2^i}\right)+n\times\left(2\left(1-\frac{1}{2^i}\right)\right)
\end{aligned}
$$

Let $i=\log_2 n$

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{n}\right)+n\times 2\left(1-\frac{1}{n}\right) \\
&= 1+2n-2 \\
&= 2n-1
\end{aligned}
$$

Thus, we have:

$$T(n)=2n-1, n\ge 1$$

Problem5 Answer:

For Merge sort pro, $T(n)=\begin{cases}3T\left(\frac{n}{3}\right)+O(n), & n>1\\ O(1), & n=1\end{cases}$

$$
\begin{aligned}
T(n) &= 3T\left(\frac{n}{3}\right)+n \\
&= 3\left[3T\left(\frac{n}{3^2}\right)+\frac{n}{3}\right]+n \\
&= 3^2T\left(\frac{n}{3^2}\right)+2n
\end{aligned}
$$

$$= 3^3 T\left(\frac{n}{3^3}\right) + 3n$$

$$= 3^i T\left(\frac{n}{3^i}\right) + i \times n$$

Let $i = \log_3 n$:

$$T(n) = nT\left(\frac{n}{n}\right) + n \log_3 n$$

$$= O(n \log n)$$

Problem6 Answer:

```
peakOfMountain(A, left, right)
    if right − low = 2
        return max(A[left], A[(left + right)/2], A[right])
    center = (left + right)/2
    if A[center−1] < A[center] and A[center] > A[center+1]
        return A[center]
    if A[center] < A[center+1]
        return peakOfMountain(A, center, right)
    if A[center] < A[center−1]
        return peakOfMountain(A, left, center)
```