Answer to the Question No. - 1

(a) For the given while loop, the loop invariant is –

At the beginning of the while loop, $k = 2^c$.

(b) Before the first iteration of the while loop, c is assigned to 0 and k is assigned to 1.

So, the base case is, $k_1 = 2^0 = 1$, which is equal to k.

Now assuming the loop invariant is true for first p iterations, so $k_p = 2^p$.

We multiply k with 2 during the iteration. It can be written that,

$K_2 = 2^1 * 2$
$K_3 = 2^2 * 2$
$K_4 = 2^3 * 2$ and so on.
Here, $k = 2^{p+1}$ and it is the same as the beginning of p+1-th iterations.

At the end: c = n, so $k = 2^n$.

(c) The runtime of the algorithm:

Algorithm	**Cost**	**Times**
function pow(*n*)		
k = 1	C_1	1
c = 0	C_2	1
while *c* < *n* **do**	C_3	n+1
k = *k* · 2	C_4	n
c = *c* + 1	C_5	n
end while		
return *k*	C_6	1
end function		

Here C is a constant and independent of the input.

We assume that each statement will take constant amount of time which is 1.

So, the runtime will be $= C_1 + C_2 + (n+1)C_3 + n\,(C_4 + C_5) + C_6$

$= C_1 + C_2 + C_3 + C_6 + n(C_3 + C_4 + C_5)$

$= a + bn = O(n)$, Where a and b are constants.

The loop invariant holds before the first iteration and if the invariant holds before an iteration then it also holds before the next one and the invariant holds at the end of the iteration. So, the algorithm is correct.

Answer to the Question No. – 2

(a) For the outer loop,

Iterations	**Value of i**
1	$i = 3n$
2	$i = 3n – 4$
3	$i = 3n – (2 * 4)$
4	$i = 3n – (3 * 4)$
….	…
k	$i = 3n – (k - 1)*4$

From the given statement, $3n – (k - 1)*4 = 1$. So, $k = (3n - 1)/4 + 1 = ¾ * n + ¾$.

For outer loop we get O(n) after discarding the constant terms.

For the inner loop,

Iterations	**Value of j**
1	$j = 20n/3^0$
2	$j = 20n/3^1$
3	$j = 20n/3^2$
4	$j = 20n/3^3$
….	…
K	$j = 20n/3^{(k-1)}$

From the given statement, $20n/3^{(k-1)} = 1$

$$\Rightarrow n = 3^{(k-1)}/ 20$$

Taking log on both side we get, $\log n = (k-1) \log 3 - \log 20$

$$\Rightarrow k = \log n / \log 3 + \log 20/\log 3 + 1$$

For inner loop we get $O(\log n)$, after discarding the constant terms.

So, we get the complexity is, $n * \log n$ that is, $\theta(n\log n)$ and $n_0 = 1$. We know that we can replace big-O with big-θ.

(b) For the outer loop,

Iterations	Value of i
1	$i = 3n^2 - 0$
2	$i = 3n^2 - 1$
3	$i = 3n^2 - 2$
4	$i = 3n^2 - 3$
….	…
K	$i = 3n^2 – (k-1)$

From the given statement, $3n^2 – (k-1) = 1$. So, $k = 3n^2$

For outer loop we get $O(n^2)$.

For the inner loop,

Iterations	Value of j
1	$j = 3n^2/2^0$
2	$j = 3n^2/2^1$
3	$j = 3n^2/2^2$
4	$j = 3n^2/2^3$
….	…
K	$j = 3n^2/2^{(k-1)}$

From the given statement, $3n^2/2^{(k-1)} = 1$

$$\Rightarrow n^2 = 2^{(k-1)}/3$$

Taking log on both side we get, $2 \log n = (k-1) \log 2 - \log 3$

$$\Rightarrow k = (2/\log 2) * \log n + \log 3 / \log 2 + 1$$

So, we get the complexity is, $n^2 * \log n$ that is, $\theta(n^2 \log n)$ and $n_0 = 1$. We know that big-O can be replaced with big-θ.

Answer to the Question No. – 3

(a) Let g and f be the function from the set of natural numbers to itself. The function f is said to be θ(g), if there are constants c1, c2 > 0 and a natural number n_0 such that $c1 * g(n) \leq f(n) \leq c2 * g(n)$ for all $n \geq n_0$.

From the given expression we get, $f(n) = 4n^5 – 50n^2 + 10n$ and $g(n) = n^5$.

Now, $4n^5 – 50n^2 + 10n \leq n^5$, (when n > 3)

$\leq 4n^5 + 10n$

$\leq 4n^5 + 10n^5$

$\leq 14n^5$ for $c = 14$ and $n_0 = 3$

That is, $4n^5 – 50n^2 + 10n = O(n^5)$

Again, $n^5 \leq n^5 – 50n^2 + 10n$, (when n > 1)

$\leq 4n^5 – 50n^2 + 10n$ for $c = 1$ and $n_0 = 1$

That is, $4n^5 – 50n^2 + 10n = \Omega(n^5)$

So we have, $4n^2 – 50n^2 + 10n \in \theta(n^5)$ is true.

(b) $o(g(n)) = \{f(n)$: for any positive constant c, there exists positive constant n_0 such that $0 \le f(n) < cg(n)$ for all $n \ge n_0\}$. So, the set of functions f(n) are strictly smaller than c*g(n), meaning that small-o notation is a stronger upper bound than big-O notation.

Intuitively, this means that as the n approaches infinity, f(n) becomes insignificant compared to g(n). In mathematical terms:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Here, $f(n) = 5n^{2/3} + 8\log n$ and $g(n) = n$. So,

$$\lim_{n\to\infty} \frac{5n^{\frac{2}{3}} + 8\log n}{n}$$
$$= \lim_{n\to\infty} \frac{5n^{\frac{2}{3}}}{n} + \lim_{n\to\infty} \frac{8\log n}{n}$$
$$= 5 * \lim_{n\to\infty} \frac{1}{n^{\frac{1}{3}}} + 8 * \lim_{n\to\infty} \frac{\log n}{n}$$
$$= 5 * 0 + 8 * \lim_{n\to\infty} \frac{\frac{1}{n}}{1}, [Applying\ L'Hospital\ rules]$$
$$= 0 + 8 * 0$$
$$= 0$$

This proves that $5n^{2/3} + 8\log n \epsilon o(n)$ is true.

(c) Let g and f be the function from the set of natural numbers to itself. The function f is said to be $\Omega(g)$, if there is a constant $c > 0$ and a natural number n_0 such that $c*g(n) \le f(n)$ for all $n \ge n_0$.

From the given expression we get, $f(n) = n^5 + 4n^2 + 15$ and $g(n) = n^3$.

According to the definition we can write, $n^5 + 4n^2 + 15 \ge 1 * n^5$; (When n>1)

$\Rightarrow 1 * n^3$ (As $n^3 < n^5$); For $c_1 = 1$ and $n_0 = 1$.

So, $n^5 - 4n^2 + 15 \epsilon \Omega(n^3)$ is true.

Answer to the Question No. – 4

Let's assume, $f1 = \log_2 n$, $f2 = 2^{\sqrt{\log_2 n}}$, $f3 = n^{1/3}$, $f4 = n^5$, $f5 = 10^n$ and $f6 = n^n$.

Comparing Asymptotic growth of function using logarithms we get,

$\log f1 = \log\log_2 n \le \log f2 = \sqrt{\log_2 n} * \log 2$; $f1 \epsilon O(f2)$

$\log f2 = \sqrt{\log_2 n} * \log 2 \le \log f3 = \frac{1}{3} * \log n$; $f2 \epsilon O(f3)$

$\log f3 = \frac{1}{3} * \log n \le \log f4 = 5 * \log n$; $f3 \epsilon O(f4)$

$$\log f4 = 5 * \log n \quad \leq \quad \log f5 = n * \log 10 \; ; \qquad f4 \in O(f5)$$

$$\log f5 = n * \log 10 \quad \leq \quad \log f6 = n * \log n \; ; \qquad f5 \in O(f6)$$

So we can write, $\log_2 n \leq 2^{\sqrt{\log_2 n}} \leq n^{1/3} \leq n^5 \leq 10^n \leq n^n$.