1.a) Need to prove: $\exists c>0$ and $n_0>0$ such that $0 \le 12n^3+11n^2+10 \le cn^3$, for all $n \ge n_0$

$0 \le 12n^3+11n^2+10 \le 12n^3+11n^3+10n^3$, for $n \ge 1$

$\le 33n^3$

$\therefore c=33,\ n_0=1$

$\therefore 12n^3+11n^2+10 \in O(n^3)$

b) Need to prove: $\exists c>0$ and $n_0>0$, such that $0 \le cn^3 \le 12n^3+11n^2+10$, for all $n \ge n_0$

$0 \le n^3 \le 12n^3+11n^2+10$, for all $n \ge 1$

$\therefore c=1,\ n_0=1$

$\therefore 12n^3+11n^2+10 \in \Omega(n^3)$

c) Need to prove: $\exists c_1, c_2>0$ and $n_0>0$, such that $0 \le c_1 n^3 \le 12n^3+11n^2+10 \le c_2 n^3$, for all $n \ge n_0$

according to a) and b), we can use the rest

$\therefore c_1=1,\ c_2=33,\ n_0=1$

$\therefore 12n^3+11n^2+10 \in \Theta(n^3)$

d) Need to prove: for all c, $\exists n_0>0$, such that $0 \le 1000n < c(n\log n)$, for all $n \ge n_0$

$1000 < c\log n$

$\frac{1000}{c} < \log n$

$e^{\frac{1000}{c}} < n$

$\therefore n_0 = \max\{1, e^{\frac{1000}{c}}\}$

$\therefore 1000n \in O(n\log n)$

e) Need to prove: for all c, $\exists n_0>0$, such that $0 \le c(n^{20}) < n^n$

$\log c + 20\log n < n\log n$

$(20-n)\log n < -\log c$

$(n-20)\log n > \log c$

2.a) $f(n) \in \omega(g(n))$

$L = \lim_{n\to\infty} \frac{\sqrt{n}}{(\log n)^2}$

$= \lim_{n\to\infty} \frac{\frac{1}{2}n^{-\frac{1}{2}}}{2\frac{\log n}{n}}$

$= \lim_{n\to\infty} \frac{\sqrt{n}}{4\log n}$

$= \lim_{n\to\infty} \frac{\frac{1}{2}n^{-\frac{1}{2}}}{\frac{4}{n}}$

$= \lim_{n\to\infty} \frac{\sqrt{n}}{8}$

$= \infty$

$\because L=\infty$

$\therefore \sqrt{n} \in \omega(\log n)^2$

b) $f(n) \in \Theta(g(n))$

$L = \lim_{n\to\infty} \frac{n^3(5+2\cos 2n)}{3n^2+4n^3+5n}$

$\because -1 \le \cos 2n \le 1$

$-2 \le 2\cos 2n \le 2$

$\therefore 3 \le 5+2\cos 2n \le 7$

$L = \lim_{n\to\infty} \frac{3n^3}{3n^2+4n^3+5n}$ to $\lim_{n\to\infty} \frac{7n^3}{3n^2+4n^3+5n}$

$= \frac{3}{4}$ to $\frac{7}{4}$

$\therefore 0 < L < \infty$

$\therefore n^3(5+2\cos 2n) \in \Theta(3n^2+4n^3+5n)$

3.a) False, consider $f(n):=n$, $g(n):=\begin{cases} 0, & \text{if odd} \\ n^2+1, & \text{if even}\end{cases}$

If $g(n)$ is odd, $f(n)=n$, $cg(n)=c$

$\therefore f(n) > cg(n)$. Therefore $f(n) \notin O(g(n))$

$f(n) \notin \omega(g(n))$

We need to prove $f(n) \notin O(g(n))$ or $f(n) \notin \Omega(g(n))$ to show $f(n) \notin \Theta(g(n))$

If $f(n) \in O(g(n))$, then, $\exists n_0>0$ and $c>0$ such that $f(n) \le cg(n)$ for all $n \ge n_0$. Since $f(n)=n$, $cg(n)=c$, $n>c$, $\therefore f(n) \notin O(g(n))$.

$\therefore f(n) \in \Theta(g(n))$

$\therefore f(n) \in O(g(n))$ and $g(n) \in \omega(g(n)) \not\Rightarrow f(n) \in \Theta(g(n))$

b) True.

4. $S(n)=\sum_{i=1}^{n}\frac{i}{2^i}=\sum_{i=1}^{n} i\left(\frac{1}{2}\right)^i$

$S(n)=\left(1\cdot\left(\frac{1}{2}\right)^1\right)+\left(2\cdot\left(\frac{1}{2}\right)^2\right)+\cdots+\left(n\left(\frac{1}{2}\right)^n\right)$

$2S(n)=1+\left(2\cdot\frac{1}{2}\right)+\left(3\cdot\left(\frac{1}{2}\right)^2\right)+\cdots+\left(n\cdot\left(\frac{1}{2}\right)^{n-1}\right)$

$2S(n)-S(n)=1+\left(\frac{2}{2}-\frac{1}{2}\right)+\left(\frac{3}{4}-\frac{2}{4}\right)+\cdots+\left(\frac{n}{2^{n-1}}-\frac{n-1}{2^{n-1}}\right)+\left(n\cdot\left(\frac{1}{2}\right)^n\right)$

$=\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{n-1}}\right)-\left(n\left(\frac{1}{2}\right)^n\right)$

$=\sum_{i=0}^{n-1}\left(\frac{1}{2}\right)^i+n\left(\frac{1}{2}\right)^n$

$=\frac{1-\left(\frac{1}{2}\right)^n}{1-\frac{1}{2}}-n\left(\frac{1}{2}\right)^n$

$=\frac{1-\left(\frac{1}{2}\right)^n}{\frac{1}{2}}-n\left(\frac{1}{2}\right)^n$

$=2-2\left(\frac{1}{2}\right)^n-n\left(\frac{1}{2}\right)^n$

$=2+(n+2)\left(\frac{1}{2}\right)^n$

5. a) 2^n

b) worst case $= n+1$

Example: $n=4$, $V=[0,0,0,0]$

$\Rightarrow$ in this case i is initially $=1$ then will be add up to 5. and from 1 to 5, print print 5 times of "Hello world!"

$\therefore$ # of calls to print $= n+1$

$5 = 4+1$

which is match to worst case: $n+1$

$\therefore$ worst case # of calls to print: $n+1$

c) S_i denote the subset of size n for which the number of calls to print is i, S_i are all the possible vector can be inputed, for a size i. For example $i=3$, then are $2^3=8$.

$S_i=2^i$

d) $\frac{1}{n}\times\sum_{i=1}^{n}2^i$

$=\frac{1}{n}\times(2^1+2^2+\cdots+2^n)$

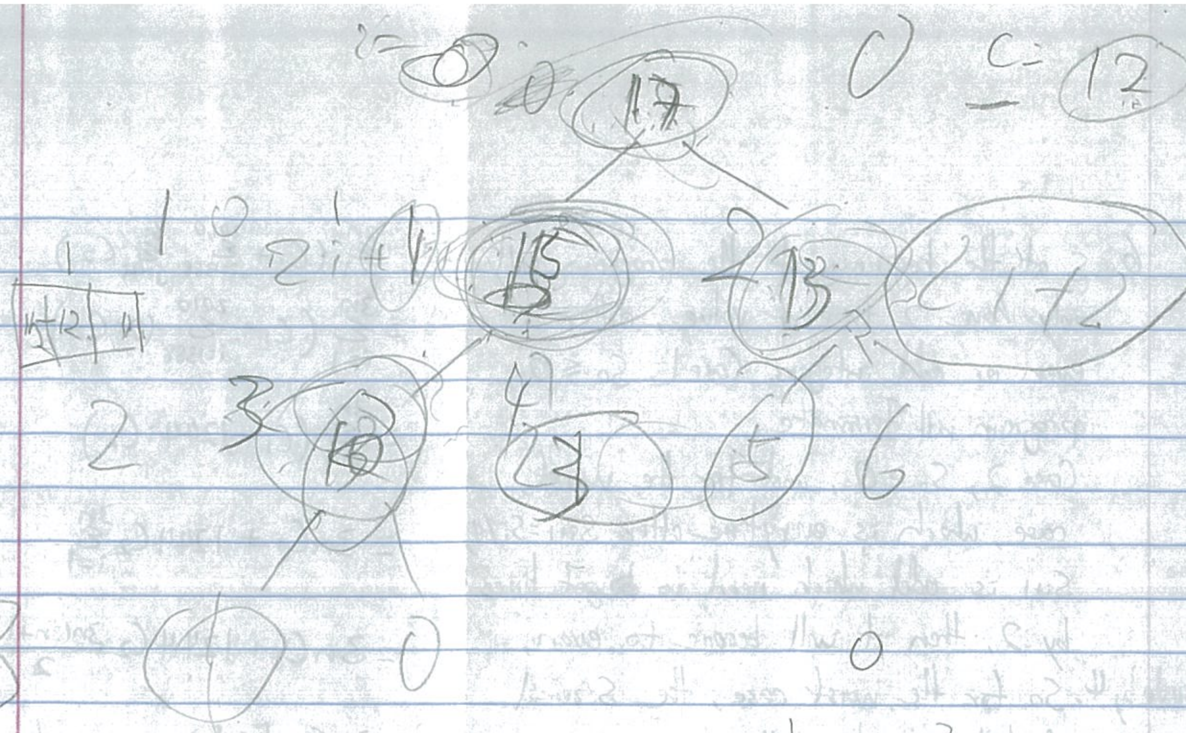
$=\frac{1}{n}\times 2(2^n-1)$

$=\frac{2}{n}(2^n-1)$

6. S at the beginning of the program will only have 2 type of value, either even or odd integers. Case1, $S_0 \le 0$, program will terminate.
Case 2, $S_0 > 0$, and for the worst case, which is everytime after $S_{i+1} = S_i/4$, S_{i+1} is odd which need to get times by 2, then it will become to even.
Then divide by 4. So for the worst case, the S_i remind at last can be written as:
$S_i = \lim_{i\to\infty} (\frac{1}{2})^i \cdot S_0 = 0$
where the program will terminate at that point.
Therefore the code fragment will always terminate.

7. $\sum_{i=1}^{3n} (C_1 + \sum_{i=1388}^{2010} \sum_{j=4i}^{6i} C_2)$
$= \sum_{i=1}^{3n} (C_1 + \sum_{i=1388}^{2010} (2iC_2))$
$= \sum_{i=1}^{3n} (C_1 + 1244iC_2)$
$= 3nC_1 + 1244C_2 \sum_{i=1}^{3n}$
$= 3nC_1 + 1244C_2 \frac{3n(3n+1)}{2} \in \Theta(n^2$

8. a) So first we convert the array to a tree structure (in algorithm not real convertion) We set the height = 0 for root. And check if it's greater or equal to c.* If yes, we print the root and call this function again by give the new height: $h = 2*h+1$ for left node and $h = 2*h+2$ for the right node. Also we need to check if h is in the range.* If no, we do nothing / return, end this line. But might have other line process after this. Therefore the program will print numbers $\ge c$ and once meet $c <$ num end recursion for 1 subprocess. Therefore the running time of my algorithm is $O(k)$, where k is the number of integers reported/outputted.



main

15	12	13	3	2	5	6