1.

a) $\begin{bmatrix} F(n+1) & F(n) \\ F(n) & F(n-1) \end{bmatrix}$ Which does this approach work

$$= \begin{bmatrix} 1*F(n)+1*F(n-1) & 1*F(n-1)+1*F(n-2) \\ 1*F(n)+0*F(n-1) & 1*F(n-1)+0*F(n-2) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(n) & F(n-1) \\ F(n-1) & F(n-2) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F(2) & F(1) \\ F(1) & F(0) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n}$$

b) Change number n to base 2, it will be shown as $a_{[\log^n+1]} \ldots a_3 a_2 a_1$ which a_i is 0 or 1.

$$\begin{bmatrix} F(n+1) & F(n) \\ F(n) & F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{a_1*2^0} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{a_2*2^1} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{a_3*2^2} \ldots \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{a_{[\log^n+1]}*n}$$

So we only need to calculate $\log^n$ matrices, each one is the square of the previous one.
The total steps are $2\log^n$, time complexity is $O(\log^n)$

2.

a) $(a+ib)(c+id)$
$= ac + ibc + iad + i^2bd$
$= ac - bd + i(bc + ad)$
let $A = (a+b)c, B = (c+d)b, C = (b-a)d$
$(a+ib)(c+id)$
$= A - B + i(B - C)$
Using 3 real numbers multiplications and 5 real numbers additions.

b) $(a+ib)^2$
$= (a+ib)(a+ib)$
$= a^2 - b^2 + i(ab + ab)$
$= (a+b)(a-b) + i(ab + ab)$
Using 2 real numbers multiplications and 3 real numbers additions

c) $(a+ib)^2(c+id)^2$
$= \left((a+ib)(c+id)\right)^2$
Using 3 real numbers multiplications we could calculate $(a+ib)(c+id) = x + iy$
Using 2 real numbers multiplications we could calculate $(x+iy)^2$
So using 5 multiplications only.

3.

a) $A = a_{17}x^{17} + a_{19}x^{19} + a_{21}x^{21} + a_{23}x^{23}$
$B = b_{17}x^{17} + b_{19}x^{19} + b_{21}x^{21} + b_{23}x^{23}$
$P(x)Q(x) = a_0b_0 + a_0B + b_0A + AB$

$a_0 b_0 \rightarrow 1\ multiplication$
$a_0 B \rightarrow 4\ multiplications$
$b_0 A \rightarrow 4\ multiplications$
$AB \rightarrow 7$ multiplication, because
$R(X) = AB = c_0 x^{34} + c_1 x^{36} + c_2 x^{38} + c_3 x^{40} + c_4 x^{42} + c_5 x^{44} + c_6 x^{46}$

$$\begin{bmatrix} {x_1}^{34} & \cdots & {x_1}^{46} \\ \vdots & \ddots & \vdots \\ {x_7}^{34} & \cdots & {x_7}^{46} \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_6 \end{bmatrix} = \begin{bmatrix} c_0 {x_1}^{34} + \cdots + c_6 {x_1}^{46} \\ \vdots \\ c_0 {x_7}^{17} + \cdots + c_6 {x_7}^{46} \end{bmatrix} = \begin{bmatrix} R(x_1) \\ \vdots \\ R(x_7) \end{bmatrix} \quad \text{7 multiplications}$$

$Total\ 1 + 4 + 4 + 7 = 16\ multiplications$

b) $R(x) = P(x)Q(x) = c_0 + c_1 x^{17} + c_2 x^{19} + c_3 x^{21} + c_4 x^{23} + c_5 x^{34} + c_6 x^{36} + c_7 x^{38} + c_8 x^{40} + c_9 x^{42} + c_{10} x^{44} + c_{11} x^{46}$

$$\begin{bmatrix} 1 & {x_1}^{17} & \cdots & {x_1}^{46} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & {x_{12}}^{17} & \cdots & {x_{12}}^{46} \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_{11} \end{bmatrix} = \begin{bmatrix} c_0 + c_1 {x_1}^{17} + \cdots + c_{11} {x_1}^{46} \\ \vdots \\ c_0 + c_1 {x_{12}}^{17} + \cdots + c_{11} {x_{12}}^{46} \end{bmatrix} = \begin{bmatrix} R(x_1) \\ \vdots \\ R(x_{12}) \end{bmatrix}$$

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{11} \end{bmatrix} = \begin{bmatrix} 1 & {x_1}^{17} & \cdots & {x_{11}}^{46} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & {x_{12}}^{17} & \cdots & {x_{12}}^{46} \end{bmatrix}^{-1} \begin{bmatrix} R(x_1) \\ \vdots \\ R(x_{12}) \end{bmatrix}$$

$x_1\ to\ x_{12}\ are\ small\ numbers, we\ could\ get\ P(x_1) \ldots P(x_{12}), Q(x_1) \ldots Q(x_{12})\ by\ linear\ time.$
$So\ we\ only\ need\ 12\ multiplications\ to\ calculate\ R(x_1) \ldots R(x_{12})$

4. $a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7$ →000 001 010 011 100 101 110 111
$a_0 a_4 a_2 a_6 a_1 a_5 a_3 a_7$ →000 100 010 110 001 101 011 111
So change the index to binary, reverse the digits.
The new index is the permutation of the leaves.

5. Let the edge length as the node value

```
int transform(Node<Integer> root) {
    if (root == null) {
        return 0;
    }
    int leftValue = transform(root.getLeft());
    int rightValue = transform(root.getRight());
    int maxValue = Math.max(leftValue, rightValue);
    if (leftValue != 0) {
        root.getLeft().setValue((int) root.getLeft().getValue() + maxValue - leftValue);
    }
    if (rightValue != 0) {
        root.getRight().setValue((int) root.getRight().getValue() + maxValue - rightValue);
    }
    return maxValue + root.getValue();
}
```

6. Sort items by polishing time by descend. Then process them one by one.
Assume the previous solution $\{i_1, i_2, i_3 \ldots i_n\}$ is not the optimal one. In order to achieve the optimal one, we should swap $i_x\ to\ i_y$ at least one time.
$i_x > i_j$, i_j would have no effect on the completion time. But i_x may cause completion time delay or have no change.

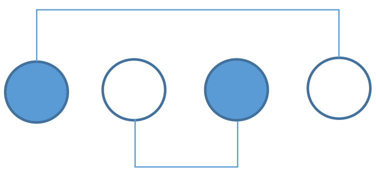
So swap i_x *to* i_y will let the solution same or worse.
Thus $\{i_1, i_2, i_3 \dots i_n\}$ is the optimal solution.

7. Start from any vertex and set it as selected area. Choose the shortest edge to the selected area. Add the edge and vertex into the selected area. Repeat until all vertices are added into selected area.

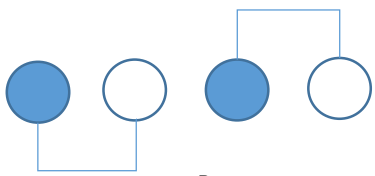
8.
   1) Set Loololong as START
   2) Set first house start from START as FIRST;
   3) Go straight until reaching the position which is 5 km far away from FIRST;
   4) Set a base station on this position;
   5) Move forward 5km and set current position as START
   6) Repeat step 2 to 5 until reaching Goolagong;

9.
   1) Find the people in the people list whose known number or unknown number is less than 5.
   2) Remove this one from people list and Pairs list.
   3) Repeat finding the people and remove him/her until everyone's known number and unknown number is greater than or equal to 5.

10.
   a) If start by connecting the closest pair of black and a white dot, Connection A would be one of the



A



B

possibility. But the total length is 4 which is larger than the optimal length 2 as connection B.
   b) The optimal solution:
      Select the left most white node and left most black node.
      Connect and ignore them.
      Repeat

11.

Create an empty **List** which length is n.
For task i = 1 to n:
    If from **List[0]** to **List[i-1]** , one or more are empty:
        Put $\mathbf{weight_i}$ into the right most empty one.
    Else:
        If $\mathbf{weight_i}$ is less than all values:
            Add $\mathbf{weight_i}$ to **totalPenalty**
        Else:
            Put $\mathbf{weight_i}$ into the one has minimum value.
            Add that minimum value to **totalPenalty**

12. BookList = [10]

```
While bookId in sequence:
    If bookId in BookList:
        Continue
    Else:
        Add bookId into BookList
        If BookList is full:
            Go to library and the books in Booklist
            Clear BookList
```

13. Select 10 longest areas which do not need to be covered and not include the first and last stall.
Then cover the other 11 parts.

14.

```
len1 = len(string1)
len2 = len(string2)
while len1 > 0 and len2 > 0:
    if string1[len1 - 1] == string2[len2 - 1]:
        romve the letter on the position (len2-1) from the string2
        len1 -= 1
        len2 -= 1
    else:
        len2 -= 1
if len1 == 0:
    return string2
if len2 == 0:
    return None
```