Ex 1. 1. The determinant can be expressed as

$$\sum_{k_1=1}^{n}\sum_{k_2=1}^{n}...\sum_{k_n=1}^{n}\epsilon_{k_1k_2...k_n}X_{1,k_1}X_{2,k_2}...X_{n,k_n}$$

where ϵ is 1 when $k_1k_2...k_n$ is an even permutation, -1 if it is an odd permutation and 0 if it is not a permutation.

If determinant is identically 0, it means that for every $X_{1,k_1}X_{2,k_2}...X_{n,k_n}$ where $k_1k_2...k_n$ is a permutation will contain a 0. Since $k_1k_2...k_n$ is a permutation of $1, 2..., n$, it means that the matching strategy $1-k_1, 2-k_2, ...n-k_n$ does not work, and in that case all matching possibilities are excluded, so there is no perfect matching.

If determinant is not identically 0, it means that for at least one permutation $k_1k_2...k_n$, $X_{1,k_1}X_{2,k_2}...X_{n,k_n}$ contains no zero, so there is a matching strategy $1-k_1, 2-k_2, ...n-k_n$.

In conclusion, the determinant is identically zero if and only if no perfect matching exists.

2. The maximum matching algorithm can be used. If the max match is the total number of vertices of a side, it means that the graph has a perfect matching.

3. The complxity should be $\mathcal{O}(|V||E|)$ and since the existence of perfect matching matches the maximum number of matches, the algorithm is correct.

4. It is useful as it is correct and has a reasonable time complexity.

Ex 2. 1. The algorithm is shown below.

Algorithm 1 Middle Node Finding

Input: A single linked list L
Output: Its middle node

$node_{slow} \leftarrow L.head$
$node_{fast} \leftarrow L.head$
while $node_{fast}! = L.tail$ **do**
    $node_{slow} \leftarrow node_{slow}.next$
    $node_{fast} \leftarrow node_{fast}.next$
    if $node_{fast} = L.tail$ **then**
        return $node_{slow}$
    end if
    $node_{fast} \leftarrow node_{fast}.next$
end while
return $node_{slow}$

2. The algorithm is shown below.

Algorithm 2 Loop Detecting

Input: A single linked list L
Output: Whether it contains a loop

$node_{slow} \leftarrow L.head$
$node_{fast} \leftarrow L.head$
while $node_{fast}! = L.tail$ **do**
  $node_{slow} \leftarrow node_{slow}.next$
  $node_{fast} \leftarrow node_{fast}.next$
  if $node_{fast} = L.tail$ **then**
    return $false$
  end if
  $node_{fast} \leftarrow node_{fast}.next$
  if $node_{fast}.next = node_{slow}$ **then**
    return $true$
  end if
end while
return $false$

Ex 3. 1. Obviously the collector should buy at least n boxes.

3. The expectation can be calculated as

$$E[X] = \sum_{k=1}^{n} \frac{n}{k} = n \sum_{k=1}^{n} \frac{1}{k} \geq n \int_{1}^{n} \frac{1}{x} dx = n \log n$$

and as $n \sum_{k=1}^{n} \frac{1}{k} \leq n \int_{1}^{n} \frac{2}{x} dx = 2n \log n$, it can be concluded that $E[X] = \Theta(n \log n)$

4. It means that the time does not grow linearly as the nubmer of coupons growing, and the time for colloecting the last few coupons will be much more if n is large.