1- Flowchart for an algorithm is given as follows:

The program finds the binary representation of the number ‘k’

k=25 can be written as $k = 25 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$, hence the binary representation of k=25 is 11001.
c(10p) - Determine the time formula for the algorithm and convert this time formula to the best possible big-O notation. *(Explain properly or you will not get any points!)*

Within the while loop, each time the number \( k \) divided by 2 until \( k \) becomes 0. So the number of the operations (\( I \)) depend on the number \( k \),

If \( k = 2 \) the program will enter the loop only twice:
\[ k = 2/2 = 1, I = 1 \]
\[ k = 1/2 = 0, I = 2, \] then the program exits from the while loop since \( k \) becomes zero.

We can calculate the number of operations as \( I = (\log_2 k) + 1 = (\log_2 2) + 1 = 2 \)

If \( k = 16 \) the program will enter the loop 5 times:
\[ k = 16/2 = 8, I = 1 \]
\[ k = 8/2 = 4, I = 2 \]
\[ k = 4/2 = 2, I = 3 \]
\[ k = 2/2 = 1, I = 4 \]
\[ k = 1/2 = 0, I = 5, \] then the program exits from the while loop since \( k \) becomes zero.

We can calculate the number of operations as \( I = (\log_2 k) + 1 = (\log_2 16) + 1 = 5 \)

for \( k = 25 \) the program will enter the loop 5 times. \( k = 25 \) is not a power of 2 but \( 2^4 < k = 25 > 2^5 \) so the number of operations is equal to 5.

So in general we can conclude that the program enters the while loop \( I = (\log_2 k) + 1 \) times, which is the number of comparisons (\( k < > 0 \)). The number of comparisons also represents the complexity of the algorithm.

\( I = (\log_2 k) + 1 \) is the time formula.

To convert this formula we need to eliminate any constants:

According to the rule of changing bases:
\[ (\log_2 k) = \frac{\log_{10} k}{\log_{10} 2} \]

\[ \log_{10} 2 = 0.301 \text{ and } \frac{1}{\log_{10} 2} = 3.321 \text{ which is a constant number.} \]

Then \( I = (\log_2 k) + 1 = 3.321(\log_{10} k) + 1 \)

Ignoring the constants, 3.321 and 1, the big-O notation will be: \( \log k \)

The program also enters the “for” loop \( I - 1 = \log_2 k \) times. This will not change big-O notation.

2(25p)- For a given integer array, you are required to determine the items that repeats itself in the next address, determine the number of repeats and for the repeating number. Draw a flowchart diagram for the algorithm and analyze for \( A = 1, 2, 3, 3, 2, 2, 5 \)

Example: If the input is the following array \( A \),

\[ 1\ 3\ 7\ 4\ 4\ 4\ 7\ 1\ 1\ 1\ 1\ 3 \]

The output of the program should be two separate arrays called \( \text{Numbers} \) and \( \text{Repeats} \):

\( \text{Numbers} = 4, 1 \) (repeating numbers are 4 and 1)
\( \text{Repeats} = 3, 4 \) (they repeat 3 and 4 times respectively)
If we analyze the algorithm for A=1,2,3,3,2,2,5:

First we create an array called Repeats and fill the array with 1.

$T = 0, K = 0$

$I = 0 \quad A[0] = A[1] \rightarrow \text{False}$


$K = T + 1 = 1$

Repeats[0] = Repeats[0] + 1 = 2$


$K = T + 1 = 2$

Repeats[0] = Repeats[0] + 1 = 2$


$T = K = 2$

The resulting arrays are: Numbers=3,2 and Repeats=2,2
3- For a linked list based on an array, the array A containing the elements of the list and the array P containing the links are given as follows:

\[
A = \{\text{null, 23, 42, 32, 49, 44, 18, null}\}
\]

\[
P = \{4, 6, 3, 1, 5, 2, -1, \text{null}\}
\]

a(15p)- Write the array A and array P after deleting ‘23’ from the linked list.

b(15p)- Write the array A and array P after inserting an element valued ‘35’ into the proper place in the original linked list (before deleting 23).


![Diagram of linked list]

The original arrays are as follows:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>null, 23, 42, 32, 49, 44, 18, null</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>4, 6, 3, 1, 5, 2, -1, null</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a- Deleting ‘23’ only requires ‘32’ to point to ‘18’ instead of ‘23’;

\[
\begin{array}{c|cccccccc}
\text{A} & \text{null} & 23 & 42 & 32 & 49 & 44 & 18 & \text{null} \\
\text{P} & 4 & -1 & 3 & 6 & 5 & 2 & -1 & \text{null} \\
\end{array}
\]

Also you can arrange ‘23’ to point to nothing (actually you do not have to worry about what 23 points to – since none of the other items points to ‘23’ you will not reach ‘23’ anymore). Trying to remove ‘23’ from array A would be a mistake, after all why do we use linked lists?

b- Inserting ‘35’ requires ‘42’ to point to ‘35’ instead of ‘32’, also now ‘35’ should point to ‘32’;

\[
\begin{array}{c|cccccccc}
\text{A} & \text{null} & 23 & 42 & 32 & 49 & 44 & 18 & 35 \\
\text{P} & 4 & 6 & 7 & 1 & 5 & 2 & -1 & 3 \\
\end{array}
\]

You also do not need to worry where to place the new number ‘35’ in array A, you should place it at the end to avoid any unnecessary operations. You should only organize the links in array P.