Open book, open notes, but please do not ask questions.
Write all answers on these sheets.

<table>
<thead>
<tr>
<th>question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maximum</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>25</td>
<td>25</td>
<td>100</td>
</tr>
</tbody>
</table>

1. Write a Python function `BiDiag` which takes as input a positive number $n$ and returns an $n$-by-$n$ matrix $A$. All diagonal elements of $A$ are 2 and all elements just above and below the diagonal are 1:

$$
A = \begin{bmatrix}
2 & 1 & 0 & \cdots & 0 & 0 & 0 \\
1 & 2 & 1 & \cdots & 0 & 0 & 0 \\
0 & 1 & 2 & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 2 & 1 & 0 \\
0 & 0 & 0 & \cdots & 1 & 2 & 1 \\
0 & 0 & 0 & \cdots & 0 & 1 & 2
\end{bmatrix}.
$$

`BiDiag` returns $A$ as a two dimensional `numpy` array.
2. Write a Python function which removes all duplicate elements of a list given on input. Call this function \texttt{RemoveDuplicates}.
If \( L = [1, 3, 1, 4, 3, 3, 2] \), \texttt{RemoveDuplicates} will return \( [1, 4, 3, 2] \).

(a) Write an \textit{iterative} version of \texttt{RemoveDuplicates}.

(b) Write a \textit{recursive} version of \texttt{RemoveDuplicates}.

3. Apply divide and conquer to compute the sum of all numbers in a list: the total sum is the sum of the first half and the sum of the second half. Give a recursive function \texttt{RecSum} using \textit{divide and conquer} which returns the sum of a list given on input.
4. The Cantor set is defined by removing the middle third of \([0,1]\) and then removing the middle third of the remaining intervals. The \(n\)-th Cantor set is obtained by executing the recursive removal \(n\) times. Cantor sets for \(n = 0, 1,\) and 2 are below:

\[
\begin{align*}
n = 0 & : [0,1] \\
n = 1 & : [0,1/3], [2/3,1] \\
n = 2 & : [0,1/9], [2/9,1/3], [2/3,7/9], [8/9,1] \\
\end{align*}
\]

(a) Write a function that returns the total length of all intervals which have been removed to form the \(n\)-th Cantor set. Complete the function definition below:

```python
def LengthCut(n,a,b):
    """
    Returns the total length of the intervals removed from the interval \([a,b]\) to form the \(n\)-th Cantor set.
    """
```

(b) Write a function that returns the lists of intervals in the \(n\)-th Cantor set. Complete the function definition below:

```python
def CantorSet(n,a,b,L):
    """
    Returns the list of intervals for the \(n\)-th Cantor set. The list is accumulated in \(L\). In the first call \(L\) is [].
    """
```
5. We use a binary tree to store a frequency table of words. The data at a node in the tree is a tuple like \((w,n)\), where the number \(n\) is the frequency of the string \(w\).

The binary tree is ordered: all words less than the word at a node in the tree are in the left branch while all other words are in the right branch of the tree.

The tree \(T\) is represented as a recursive triple of triplets: as \((\text{left},(w,n),\text{right})\) where left and right are again trees. The empty tree is the empty tuple ()

(a) Give a Python function \(\text{LookUp}\) that given a tree and a word returns the corresponding frequency count stored in the tree. If the word does not occur in \(T\), zero must be returned. Write a \textit{recursive} version of \(\text{LookUp}\) below.

(b) \textit{Use a stack} to write an \textit{iterative} version of the recursive \(\text{LookUp}\).