Part 1: Multiple choice questions (25%)
These are in a separate quiz on Canvas.

Part 2: Hash tables
Create a hash table that has the default python list ([]) as a Bucket. You are allowed to use all inbuilt list operations in the bucket. This hash table will only contain one data variable called key unlike PA5.

- **(10%) *insert(key)***
  - Adds the key to the hash table if it doesn't already exist in the hash table
    - You do **not** need to resize/rebuild the bucket_list
  - Do nothing otherwise
- **(10%) *contains(key)***
  - Returns **True** if the hash table contains the key
  - Returns **False** otherwise
- **(5%) *remove(key)***
  - Removes the key from the hash table if it exists
  - Otherwise do nothing

Part 3: Prefix parsing tree
Finish implementing the operation ***calculate_value*** into the class PrefixParsingTree.

- **(25%) *calculate_value()***
  - Returns an integer which is the calculated value of the ***root*** of the prefix parsing tree.
  - Each node has a token, stored in a string.
    - The token of a leaf is an integer.
    - The token of a non-leaf node is an operator, plus ("+") or minus ("-")
  - The calculated value of each non-leaf node in the tree is the result of applying the operator of that node to the calculated values of its two children.
  - The calculated value of a leaf is the integer value of the number represented in the token string.
- *Operations for building and populating the tree have already been implemented. Students only need to calculate the value*.

Part 4: Bullet list tree

- **(10%) parse_bracket_file(filename)**
  - Takes in the name of the bracket file as a parameter
  - Returns whatever data structure you choose as the entry point to your tree.
    - Could be a tree class
    - Could be a node (root)
  - The bracket file can look like this:
    - **{{Thing 1}{Thing 2{Subthing 1}{Subthing 2}}}**
    - ^ Root is always empty: **{<root>{child{grandchild}}}**
    - Can go deeper, to any level

- **(10%) write_bulleted_file(filename, my_tree)**
  - Takes in the name of the output file as a parameter
  - Takes as the second parameter whatever data structure you choose as the entry point to your tree.
    - Could be a tree class
    - Could be a node (root)
    - *Same as* **parse_bracket_file()** *returns*.
  - Writes the text value of each node in a separate line.
  - Starts each line with a number of TAB equal to the depth of the node in the tree (root is depth 0 and has no value). ***TAB is "\t" in a string***.

- **(5%) write_labelled_file(filename, my_tree)**
  - Same as ***write_bulleted_file(my_tree)***.
  - In addition
    - Each first level bulletin is numbered before the ***"\t"***
      - "1. "
      - "2. "
      - "3. "
    - Each second level bulletin is lettered before the second ***"\t"***
      - " a) "
      - " b) "
      - " c) "
    - All other bulletins have a dash before the last ***"\t"***
      - " - "
      - " - "
- See examples of input (bracket) and output (bullet and label) files in exam ZIP.
- *If you don't manage to parse the bracket file you can still hard code a tree and write the operations to output bulleted and/or labelled files*.