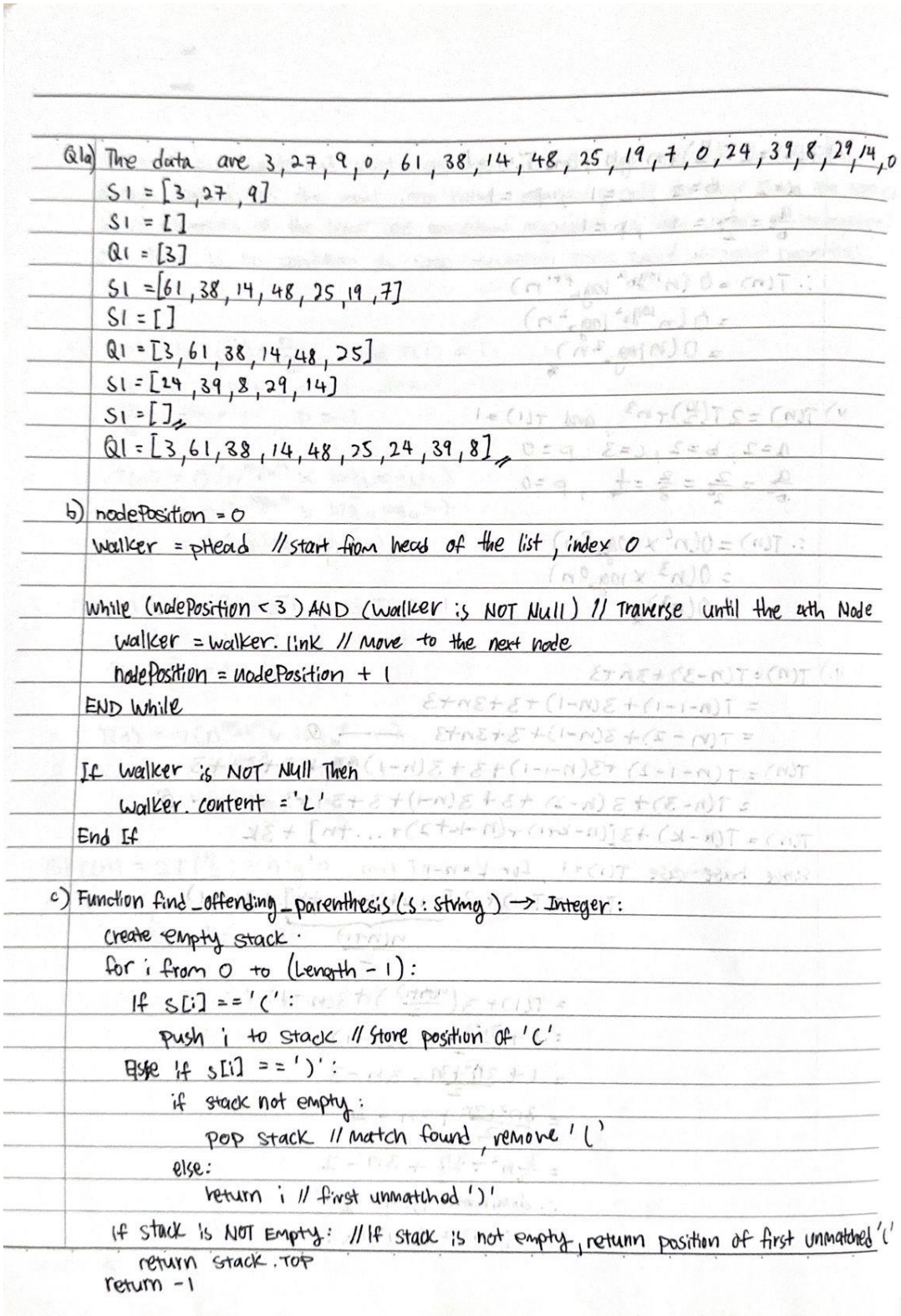
CSCI203: Algorithms and Data Structures

Assignment 2 Solutions

Question 1a, 1b and 1c:



Q1a) The data are 3, 27, 9, 0, 61, 38, 14, 48, 25, 19, 7, 0, 24, 39, 8, 29, 14, 0

S1 = [3, 27, 9]
S1 = []
Q1 = [3]
S1 = [61, 38, 14, 48, 25, 19, 7]
S1 = []
Q1 = [3, 61, 38, 14, 48, 25]
S1 = [24, 39, 8, 29, 14]
S1 = []
Q1 = [3, 61, 38, 14, 48, 25, 24, 39, 8]

b)
```
nodePosition = 0
Walker = ptrHead  // start from head of the list, index 0

While (nodePosition < 3) AND (Walker is NOT Null) // Traverse until the 2th Node
    Walker = Walker.link // Move to the next node
    nodePosition = nodePosition + 1
END While

If Walker is NOT Null Then
    Walker.content = 'L'
End If
```

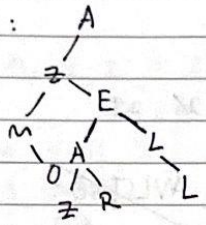
c)
```
Function find_offending_parenthesis(s: string) -> Integer:
    create empty stack.
    for i from 0 to (Length - 1):
        If s[i] == '(':
            push i to stack // store position of '('
        Else if s[i] == ')':
            if stack not empty:
                pop stack // match found, remove '('
            else:
                return i // first unmatched ')'
    If stack is NOT Empty: // If stack is not empty, return position of first unmatched '('
        return stack.top
    return -1
```

Question 2a, 2b, 2c and 2d:

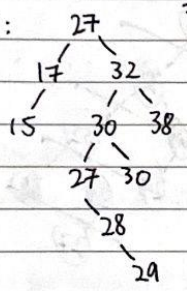
2a) Inorder : M, O, Z, Z, A, R, E, L, L, A

Preorder: A, Z, M, O, E, A, Z, R, L, L

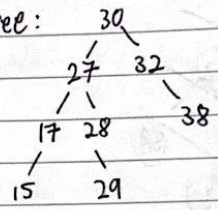
Binary tree :



2b) BST :



AVL Tree :



2c) The worst-case time complexity for building with n nodes is $O(n^2)$, because if the tree becomes skewed, each insertion takes $O(n)$ time, leading to a total construction time of $O(n^2)$.

2d) The worst case ~~occurs~~ time complexity for building an AVL Tree with n nodes is $O(n \log n)$ because each insertion takes $O(\log n)$ time, and inserting n elements results in $O(n \log n)$

2e) ~~def is_subtree(B, T):~~

~~if T is None:~~

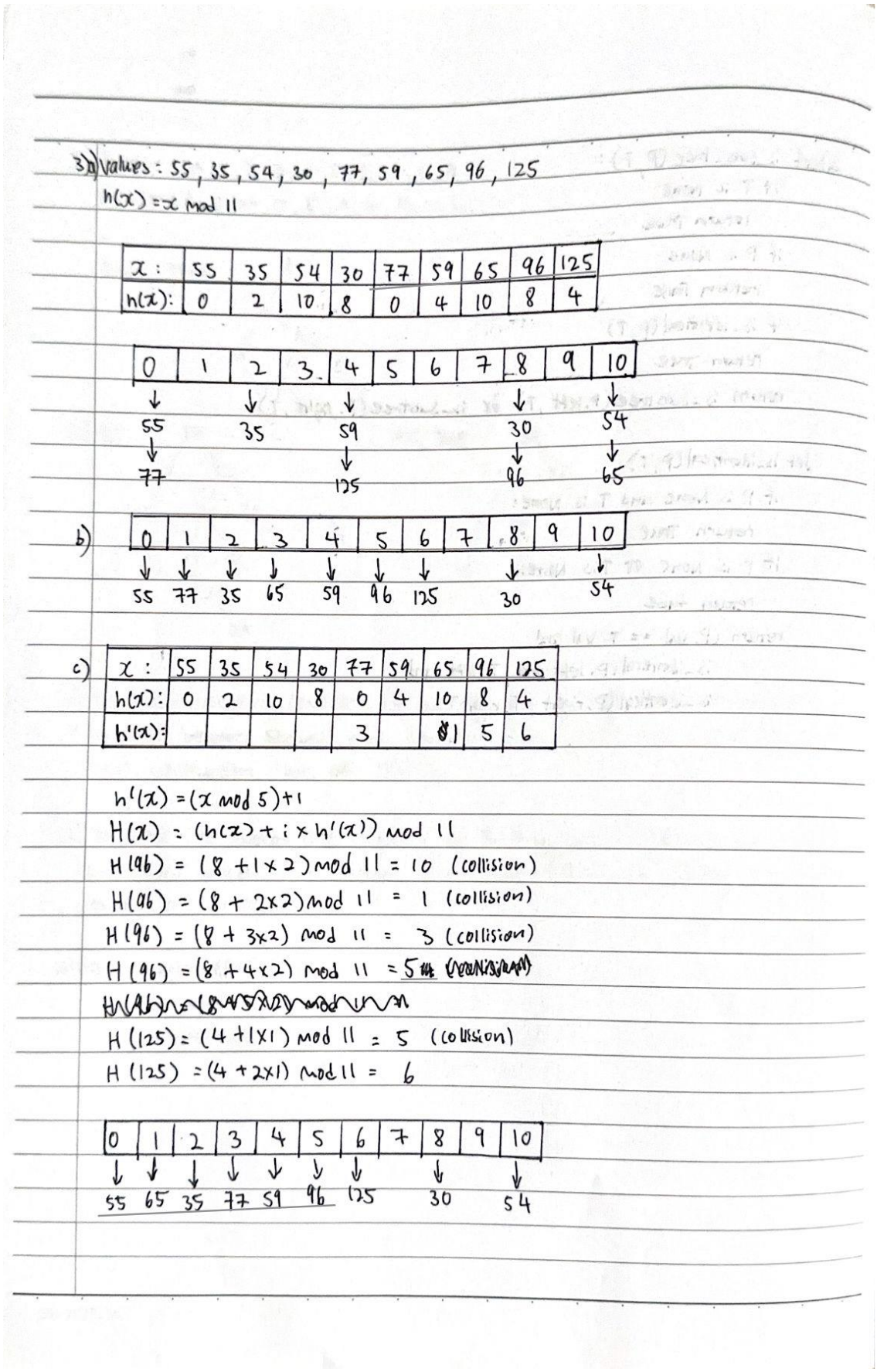
~~return True~~

Question 2e:

```
2e) def is_sub_tree(P, T):
      if T is None:
        return True
      if P is None:
        return False
      if is_identical(P, T)
        return True
      return is_subtree(P.left, T) or is_subtree(P.right, T)

    def is_identical(P, T)
      if P is None and T is None:
        return True
      if P is None or T is None:
        return false
      return (P.val == T.val and
              is_identical(P.left and, T.left and
              is_identical(P.right, T.right))
```

Question 3a, 3b and 3c:



3a) values: 55, 35, 54, 30, 77, 59, 65, 96, 125

$h(x) = x \bmod 11$

x:	55	35	54	30	77	59	65	96	125
h(x):	0	2	10	8	0	4	10	8	4

0	1	2	3	4	5	6	7	8	9	10
↓		↓		↓				↓		↓
55		35		59				30		54
↓				↓				↓		↓
77				125				96		65

b)

0	1	2	3	4	5	6	7	8	9	10
↓	↓	↓	↓	↓	↓	↓		↓		↓
55	77	35	65	59	96	125		30		54

c)

x:	55	35	54	30	77	59	65	96	125
h(x):	0	2	10	8	0	4	10	8	4
h'(x):					3		81	5	6

$h'(x) = (x \bmod 5) + 1$

$H(x) = (h(x) + i \times h'(x)) \bmod 11$

$H(96) = (8 + 1 \times 2) \bmod 11 = 10$ (collision)

$H(96) = (8 + 2 \times 2) \bmod 11 = 1$ (collision)

$H(96) = (8 + 3 \times 2) \bmod 11 = 3$ (collision)

$H(96) = (8 + 4 \times 2) \bmod 11 = 5$ (collision)

$H(125) = (4 + 1 \times 1) \bmod 11 = 5$ (collision)

$H(125) = (4 + 2 \times 1) \bmod 11 = 6$

0	1	2	3	4	5	6	7	8	9	10
↓	↓	↓	↓	↓	↓	↓		↓		↓
55	65	35	77	59	96	125		30		54

Question 3d:

3d) $h'(x,i) = (h(x) + 0.5i + 0.5i^2) \bmod 11$

x:	55	35	54	30	77	59	65	96	125
$h(x)$:	0	2	10	8	0	4	10	8	4
$h'(x,i)$:					1		5	9	7

$h'(77,1) = (0 + 0.5 + 0.5) \bmod 11 = 1$

$h'(65,1) = (10 + 0.5 + 0.5) \bmod 11 = 60$ (collision)

$h'(65,2) = (10 + 1 + 2) \bmod 11 = 2$ (collision)

$h'(65,3) = (10 + 1.5 + 4.5) \bmod 11 = 5$

$h'(96,1) = (8 + 0.5 + 0.5) \bmod 11 = 9$

$h'(125,1) = (4 + 0.5 + 0.5) \bmod 11 = 5$ (collision)

$h'(125,2) = (4 + 1 + 2) \bmod 11 = 7$

0	1	2	3	4	5	6	7	8	9	10
55	77	35		59	65		125	30	96	54

Question 4:

Screenshot of the output