COM2067/COM267 10.1.22 120 m. Final Exam

Name:
Student ID:

**Answer 10 questions.**

**1.** (10 p) Insert the following integer keys into the hash table using the hash function **h=k mod 10**. Use the following two different methods.

76, 18, 89, 48, 26

a-) Double Hashing

$h(k, i) = [h_1(k) + i h_2(k)] \bmod m \quad h_1 = (k \bmod 10)$ and $h_2 = (k \bmod 8)$

b-) Linear Probing

a-)

| | |
|---|---|
| 0 | 48 |
| 1 | |
| 2 | 26 |
| 3 | |
| 4 | |
| 5 | |
| 6 | 76 |
| 7 | |
| 8 | 18 |
| 9 | 89 |

| k | h1 | h2 |
|---|---|---|
| 76 | 6 | 4 |
| 14 | 8 | 2 |
| 89 | 9 | 1 |
| 48 | 8 | 0 |
| 26 | 6 | 2 |

b-)

| | |
|---|---|
| 0 | 48 |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | 76 |
| 7 | 26 |
| 8 | 18 |
| 9 | 89 |

**2.** (10 p) Insert the following integer keys into a red-black tree. Show the red-black tree after each insertion.

195, 46, 241, 116, 120, 175, 65

**3.** (10 p) Insert the following integer keys into an AVL tree. Show the AVL tree after each insertion.

65, 75, 20, 16, 51, 46, 25

Then, delete keys 20 and 25 from the AVL tree. Show the AVL tree after each deletion.

**4.** (10 p) Apply the following insert and delete operations on B tree (m=5). Show B tree after each operation.

Insert 175, 120, 116, 151, 146, 125, 149, 159, 181, 133, 123, 150

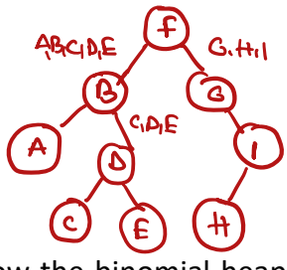
Delete 125, 146

Insert 137, 97

**5.** (10 p) Construct a binary tree from the following traversal results.

In-Order Traversal: A B C D E F G H I L, root, R

Pre-Order Traversal: F B A D C E G I H Root, L, R




**6.** (10 p) Insert the following integer keys into a binomial-heap. Show the binomial-heap after each insertion.

75, 20, 16, 51, 46, 25, 49, 59, 81, 33, 23, 50

Extract the node with the minimum key from that binomial-heap. Draw the produced binomial heap after this operation.

Binomial heap kaldı

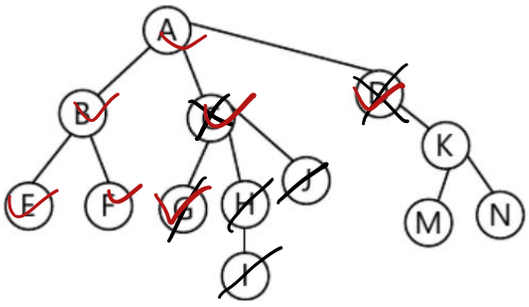
**7.** (10 p) Apply the following insert and delete operations on binary search tree (BST). Show the binary search tree after each operation.

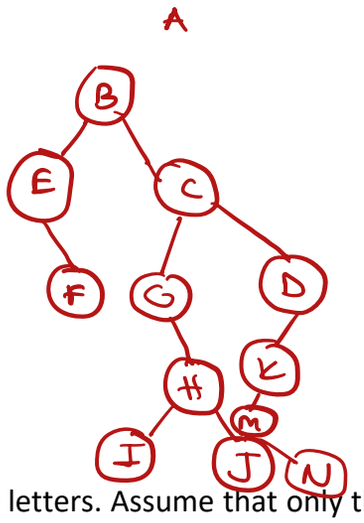
Insert 145, 38, 185, 141, 228

Delete 185, 228

Insert 190, 160, 181

**8.** (10 p) Convert the given general tree into a binary tree.




**9.** (10 p) Create a Huffman tree using the following uppercase letters. Assume that only those letters are used in the text. Given values represent the estimated frequencies of each letter in a paragraph. Using the created Huffman tree, specify a binary coding for each letter.

| M | N | O | U | I | E | A |
|---|---|---|---|---|---|---|
| 7 | 8 | 9 | 26 | 45 | 75 | 95 |

**10.** (10 p) Insert the following integer keys into max-heap. Show the max-heap after each insertion.

165, 149, 181, 127, 188, 145, 136

**11.** (10 p)

```
struct node {
  int data;   struct node * leftChild;   struct node * rightChild;   };
```

a) (5 p) Write pseudo-code of the algorithm (or C code) to calculate the total number of external nodes in a binary search tree.

**int totalExternalNodes(struct node * tree)**

b) (5 p) Write a function (C code) to print all the data values of nodes in a binary search tree using pre-order walk.

**void printPreOrder(struct node * tree)**

**12.** (10 p) Complete the definition of following function using the given node structure.

```
struct node {
  int data;   struct node * parent;   struct node * leftChild;   struct node * rightChild;  };
void insert_case3(struct node *n)
{
struct node *u, *g;  u = uncle (n);   g = grand_parent(n);
if ((u != NULL) && (u -> colour == RED))
{      n→parent→colour = BLACK
       U→colour = BLACK
}      g→colour = RED ;
else {  insert_case4(n); }  case1(g)
}
```

u kırmızıysa

1. (10 p) Insert the following integer keys into the hash table using the hash function **h=k mod 10**. Use the following two different methods.
76, 18, 89, 48, 26

a-) Double Hashing

$h(k, i) = [h_1(k) + i h_2(k)] \bmod m \quad h_1 = (k \bmod 10)$ and $h_2 = (k \bmod 8)$

i → 0 'dan 9'a kadar

b-) Linear Probing

mod 10

a)

| | |
|---|---|
| 0 | 26 |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | 76 |
| 7 | |
| 8 | 18 |
| 9 | 89 |

76 % 10 = 6
18 % 10 = 8
89 % 10 = 9
48 % 10 = 8 X

$h(48,0) = [8 + 0] \bmod 10 = 8$
$h(48,1) = [8 + 1.0] \bmod 10 = 8$
$h(48,2) = [8 + 2.0] \bmod 10 = 8$

48 yerleştirilmez

26 % 10 = 6 X

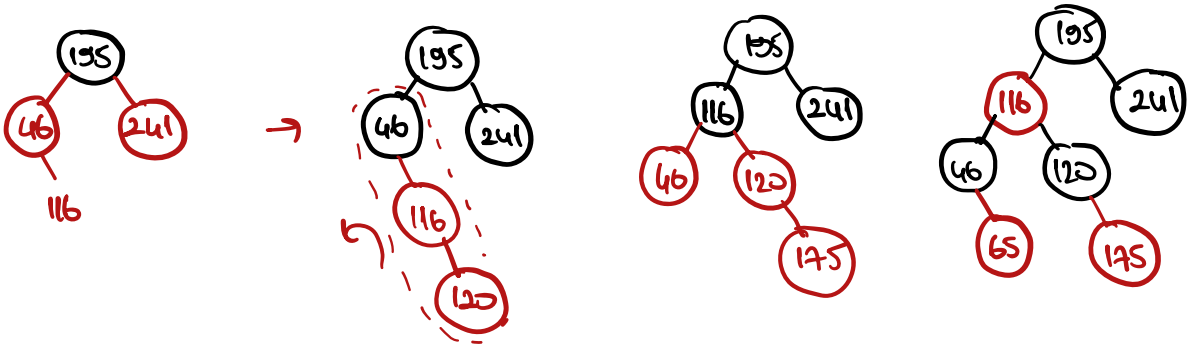
$h(26,0) = [6 + 0.2] \bmod 10 = 6$
$h(26,1) = [6 + 1.2] \bmod 10 = 8$ X
$h(26,2) = [6 + 2.2] \bmod 10 = 0$ ✓

b)

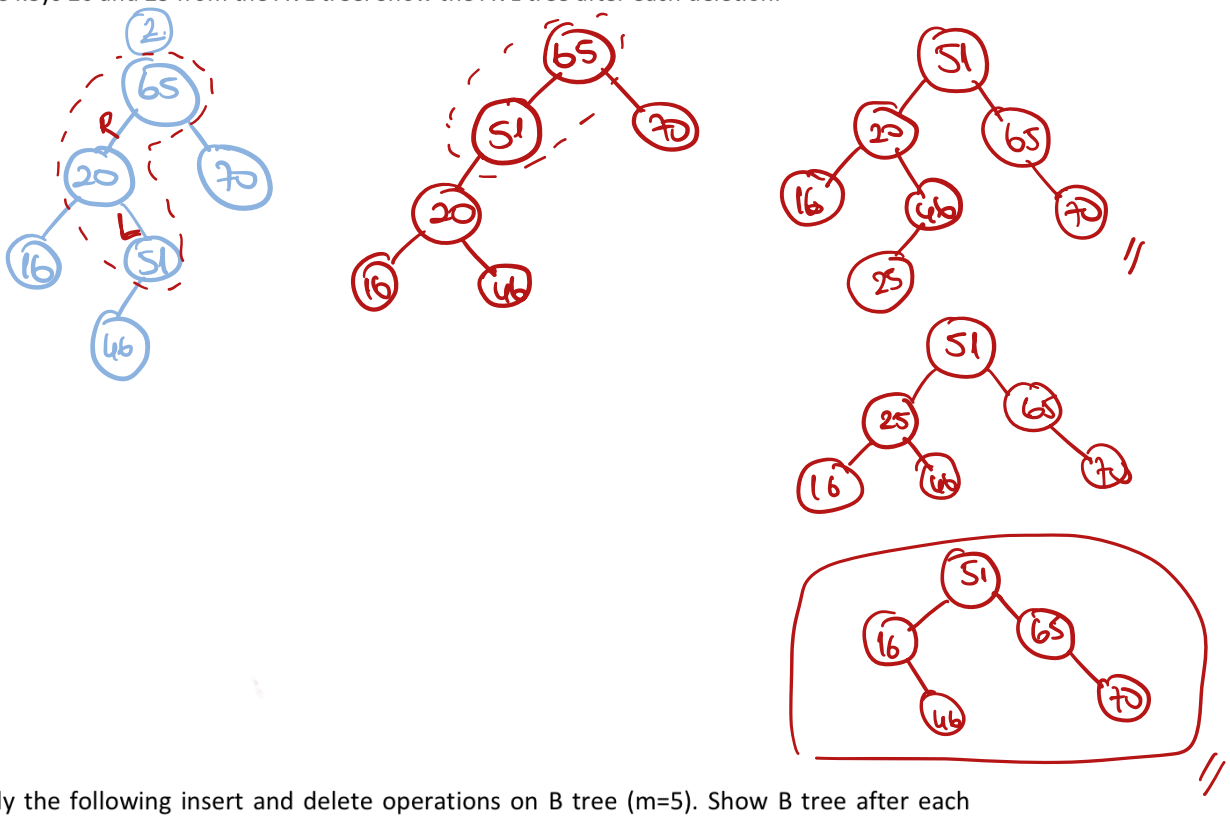
| | |
|---|---|
| 0 | 48 |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | 76 |
| 7 | 26 |
| 8 | 18 |
| 9 | 89 |

76 % 10 = 6
18 % 10 = 8
89 % 10 = 9
48 % 10 = 8
26 % 10 = 6

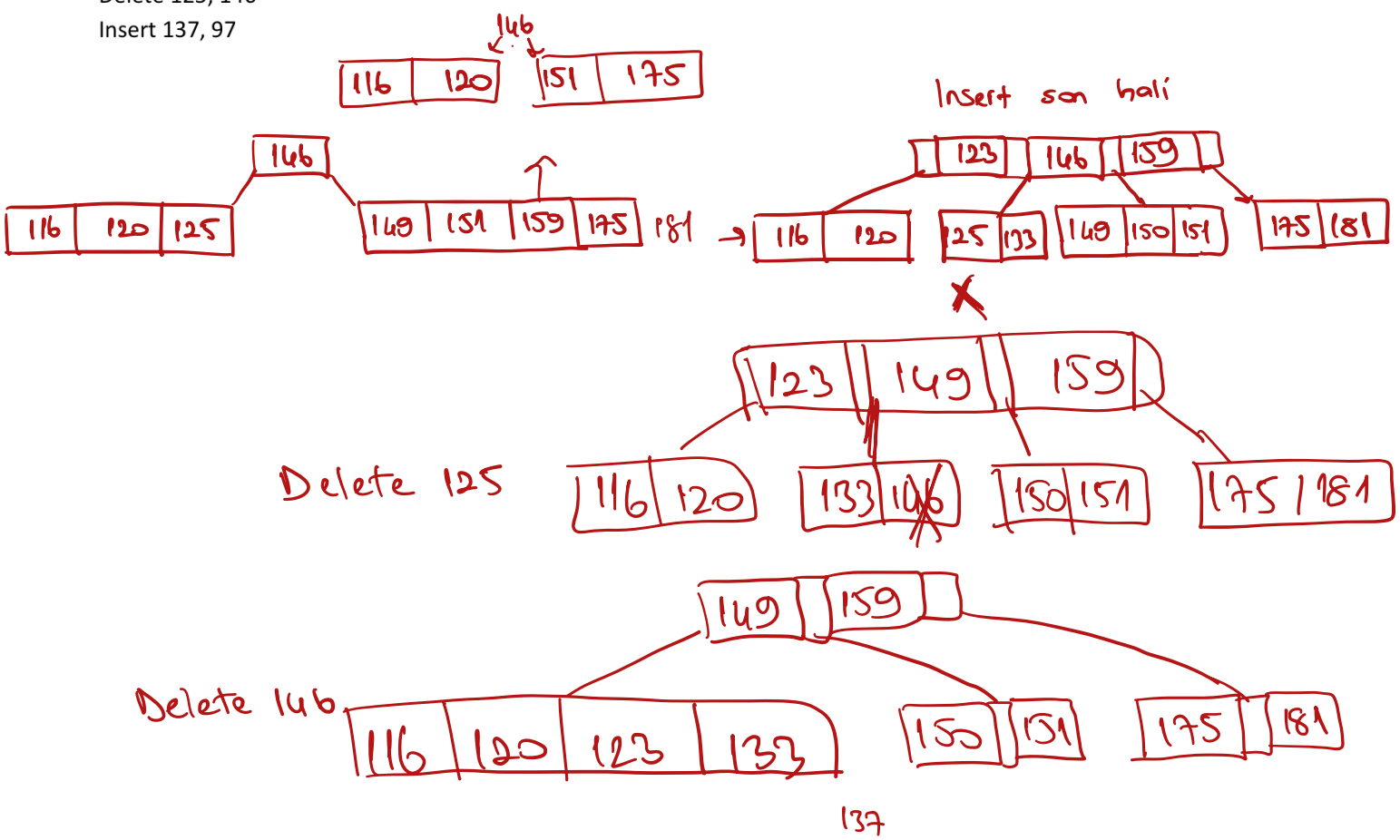
2. (10 p) Insert the following integer keys into a red-black tree. Show the red-black tree after each insertion.
195, 46, 241, 116, 120, 175, 65



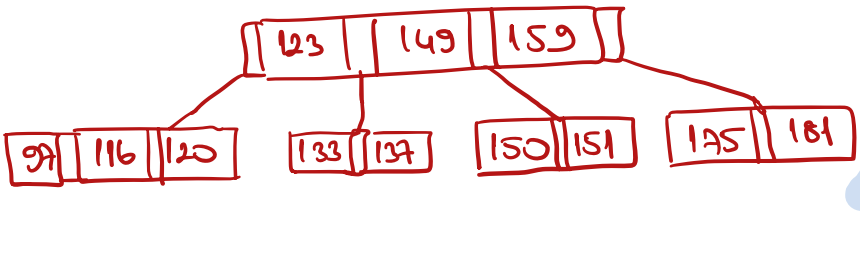
**3.** (10 p) Insert the following integer keys into an AVL tree. Show the AVL tree after each insertion.
65, 75, 20, 16, 51, 46, 25
Then, delete keys 20 and 25 from the AVL tree. Show the AVL tree after each deletion.



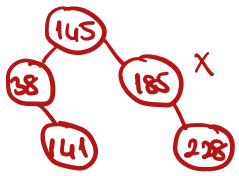

**4.** (10 p) Apply the following insert and delete operations on B tree (m=5). Show B tree after each operation.
Insert 175, 120, 116, 151, 146, 125, 149, 159, 181, 133, 123, 150
Delete 125, 146
Insert 137, 97

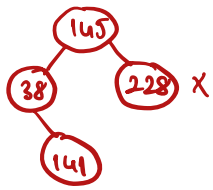


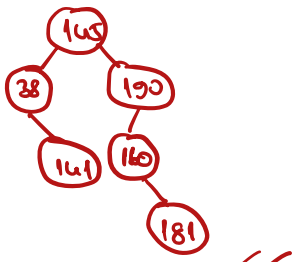
Insert 137, 97



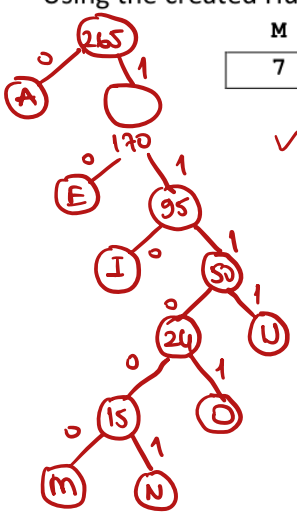

7. (10 p) Apply the following insert and delete operations on binary search tree (BST). Show the binary search tree after each operation.
Insert 145, 38, 185, 141, 228
Delete 185, 228
Insert 190. 160. 181




9. (10 p) Create a Huffman tree using the following uppercase letters. Assume that only those letters are used in the text. Given values represent the estimated frequencies of each letter in a paragraph. Using the created Huffman tree, specify a binary coding for each letter.

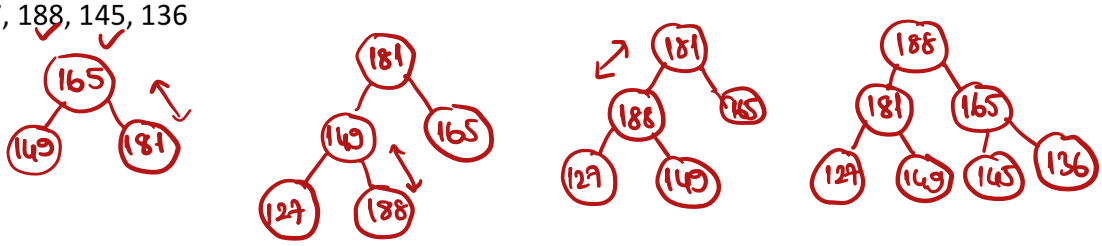



| M | N | O | U | I | E | A |
|---|---|---|---|---|---|---|
| 7 | 8 | 9 | 26 | 45 | 75 | 95 |

m → 111000
N → 111001
O → 11101
U → 1111
I → 110
E → 10
A → 0

**10.** (10 p) Insert the following integer keys into max-heap. Show the max-heap after each insertion.

165, 149, 181, 127, 188, 145, 136




**11.** (10 p)

struct node {

int data; struct node * leftChild; struct node * rightChild; };

a) (5 p) Write pseudo-code of the algorithm (or C code) to calculate the total number of external nodes in a binary search tree.

**int totalExternalNodes(struct node * tree)**

b) (5 p) Write a function (C code) to print all the data values of nodes in a binary search tree using pre-order walk.

**void printPreOrder(struct node * tree)**

11.a)

```
int totalExternalNodes(struct node* tree)
if tree is empty:
    return 0;
else if both leftChild and rightChild are NULL:
    return 1
else:
    leftcount = totalExternalNodes(tree.leftChild)
    rightcount= totalExternalNodes(tree.rightChild)
    return leftcount+rightcount
```

11. b)

```
struct node {
    int data;
    struct node * leftchild;
    struct node* rightchild; }

void printPreorder (struct node* tree) {
    if (tree != NULL) {
        printf ("%d", tree->data);
        printPreorder (tree->leftchild);
        printPreorder (tree->rightchild) }
```