EE1-08

Algorithms and Data Structures
Solutions
June 2014

1. [Calculation for a new example]

a)

The correct code is, where the corrections of the given code are highlighted.

```
int calculateF(int N){
    int result = 0;
    for (int i=0; i<=N; i++){
        if (i<=5)
            result = result + i;
        else
            result = result + 2;
    }
    return result;
}
```

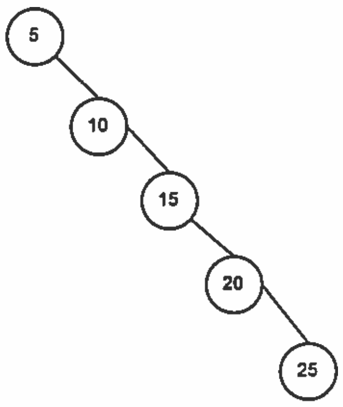
[6]

b)

```
int calculateFR(int n){
    if (n==0)
        return 0;
    else if (n<=5)
        return calculateFR(n-1)+n;
    else
        return calculateFR(n-1)+2;
}
```

[6]

c) i)

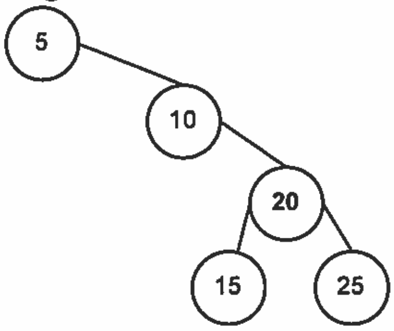


[2]

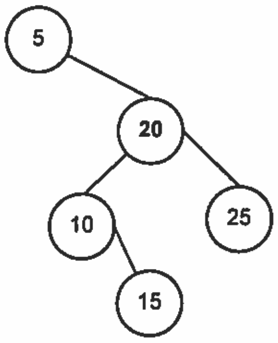
c) ii)

The tree is unbalanced. The steps are:

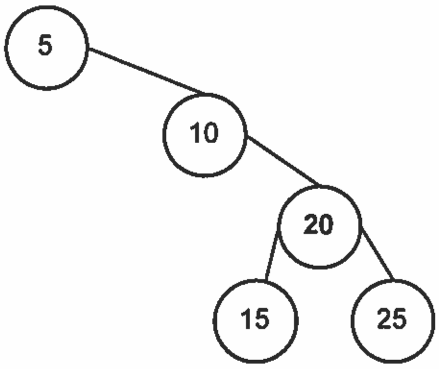
Single rotation:

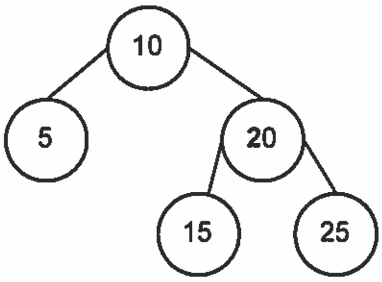


Single rotation



double rotation





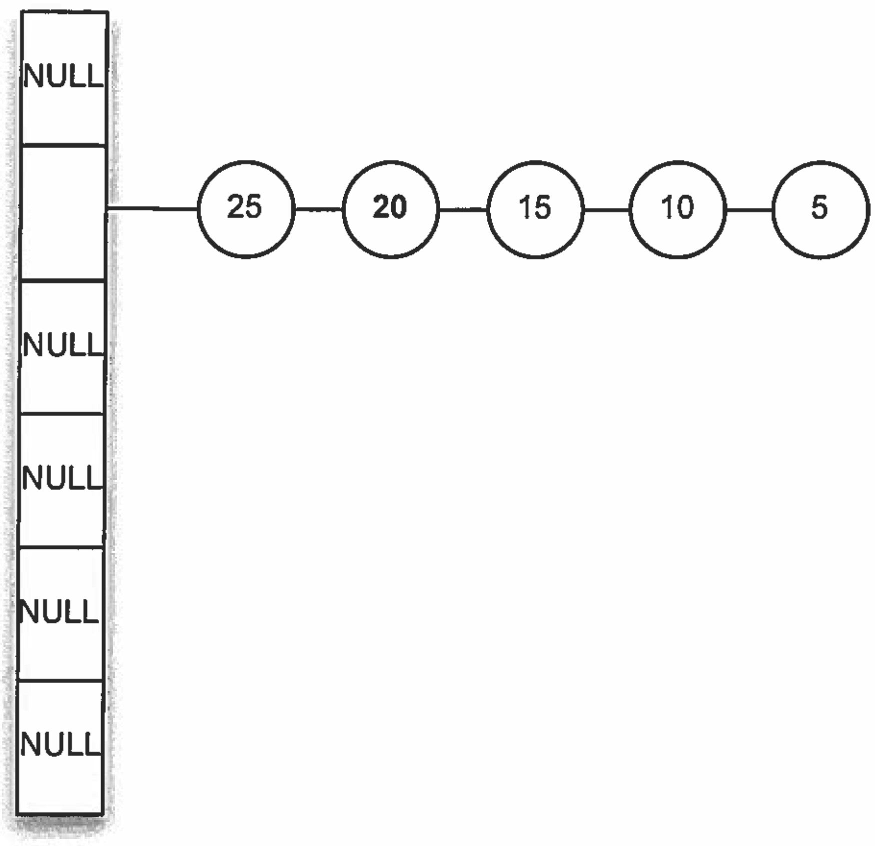
[6]

c) iii)

The height of the tree is 2

[2]

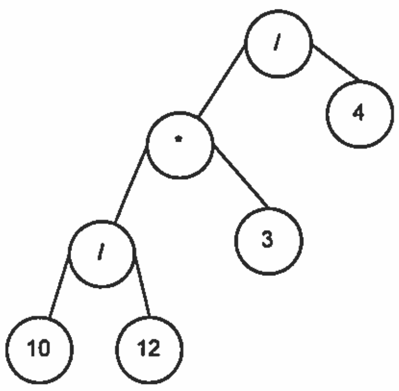
c) iv)



The selected hash function puts all the elements in the same entry of the hash table reducing its efficiency. Also, the selected function is not able to address all the entries of the table.

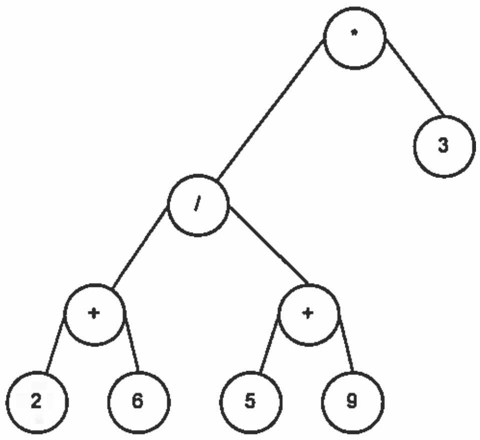
[2]

d) i)



[2]

d ii)



[2]

e)

At A: x has a random value, y = 10   (2 marks)
At B: x has a random value, y = 0   (2 marks)

There is a memory leak as p1 points at *p2, without deleting the reserved space. (1 mark)

[5]

f) i)

```
void calcProd(NodePtr hdList){
    while (hdList != NULL){
        hdList->data = 2 * hdList->data;
        hdList = hdList->next;
    }
}
```

[3]

f) ii)

```
void LinkToFirstElem(NodePtr hdList){
    if (hdList == NULL)
      return;

    NodePtr temp = hdList;
    while (hdList->next != NULL){
        hdList = hdList->next;
    hdList-> next = temp;
}
```

The problem now is that there is no indication on what is the last element of the list. (1 mark)

[4]

2) [New application]

a)

```
struct treeNode{
    int data;
    treeNode * left;
    treeNode * right;
};

typedef treeNode * TNodePtr;
```

[5]

b)

```
void largestNumber(TNodePtr hdTree, int & maxNumber) {
    if (hdTree != NULL){
        maxNumber=hdTree->data;
        largestNumber(hdTree->right, maxNumber);
    }
}
```

Note: The students should notice that since this is an ordered tree the largest element is the one at far right.

[10]

c)

```
void constructList(TNodePtr hdTree, LNodePtr & hdList) {
    LNodePtr temp = NULL;

    if (hdTree != NULL) {

      //add an element in the list
      temp = new listNode;
      temp->data = hdTree->data;
      temp->next = hdList;
      hdList = temp;

      // cont the search
      constructList(hdTree->left, hdList);
      constructList(hdTree->right, hdList);
    }
}
```

[10]

d)

```
void constructListP(TNodePtr hdTree, LNodePtr & hdList) {
    LNodePtr temp = NULL;

    if (hdTree != NULL) {

      // cont the search
      constructListP(hdTree->left, hdList);

         //add an element in the list
      temp = new listNode;
      temp->data = hdTree->data;
      temp->next = hdList;
      hdList = temp;

     constructListP(hdTree->right, hdList);

    }
}
```

[10]

e)

Two procedures/functions are needed.

```
void UpdateFields(TNodePtr hdTree){
    if (hdTree!=NULL){
        hdTree->leftNumNodes = NumberOfNodes(hdTree->left);
        hdTree->rightNumNodes = NumberOfNodes(hdTree->right);
    }
}

int NumberOfNodes(TNodePtr hdTree){
    if (hdTree==NULL)
        return 0;
    else
        return NumberOfNodes(hdTree->left) +
NumberOfNodes(hdTree->right) + 1;
}
```

[10]

f)

Here, we exploit the linked list to find the median. An easier solution can be given by using the Tree structure.

```
void medianValue(LNodePtr hdlist, float & median){
    // Count the number of elements in the list, and then find
the median.
    int count = 0;
    LNodePtr temp;
    temp = hdlist;
    while (temp != NULL){
        count++;
        temp = temp->next;
    }

    temp = hdlist;
    if (count % 2 == 1){
        for (int i=0; i<count/2; i++) {
            temp = temp->next;
        }
        median = temp->data;
    }
    else {
        for (int i=0; i<(count/2-1); i++) {
            temp = temp->next;
        }
        median = (temp->data + temp->next->data)/2.0;
    }

}
```

[15]