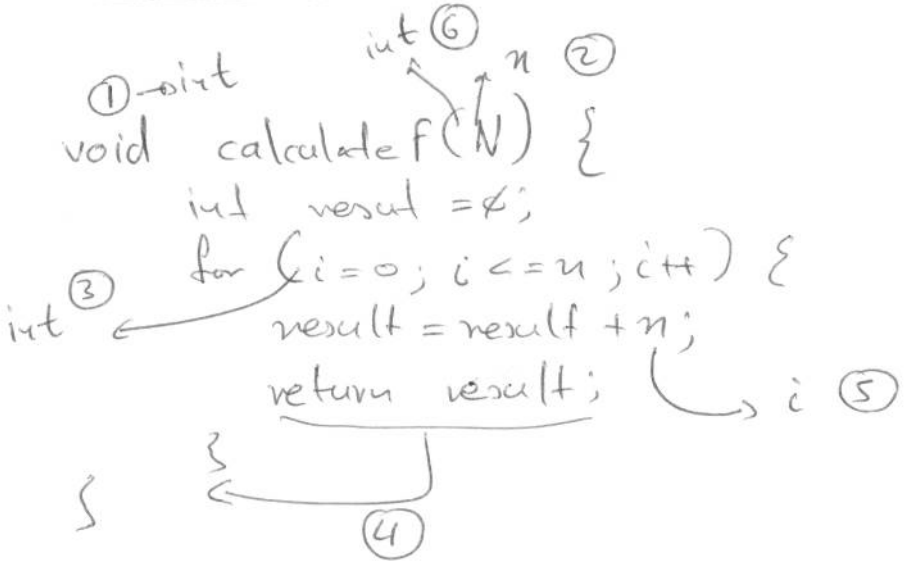
Solutions ~~Solutions~~ 2013

1.a)

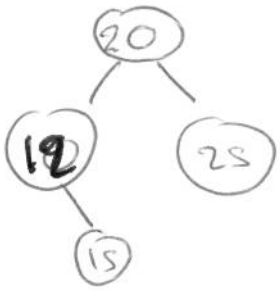


[6]

b)

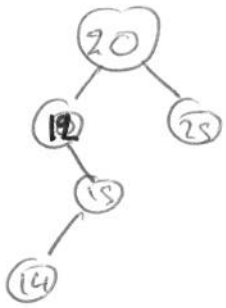
```
int calculateFR (int n) {

    if (n == ∅)
        return ∅;
    else
        return calculateFR(n-1) + n;
}
```

c) i)

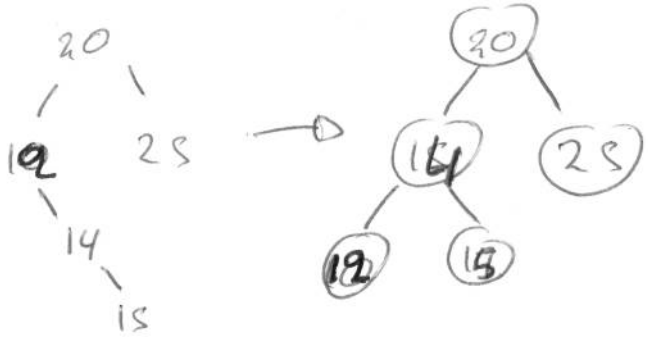


ii) The tree in part (i) is balanced as all the BFs are within the range $\in [-1, 1]$.

iii)



iv) The tree is not balanced. Resulting tree. (double rotation).

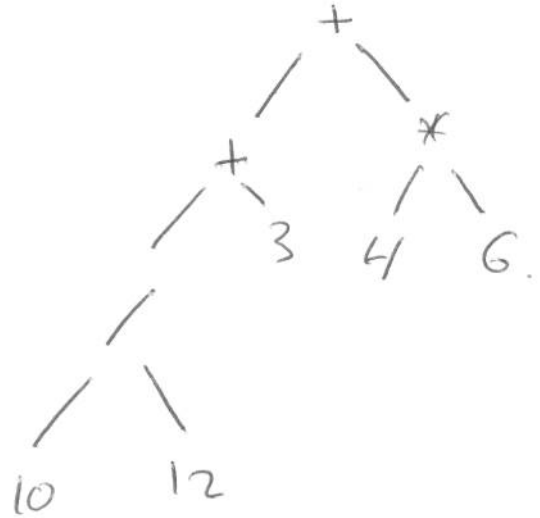


v) 2

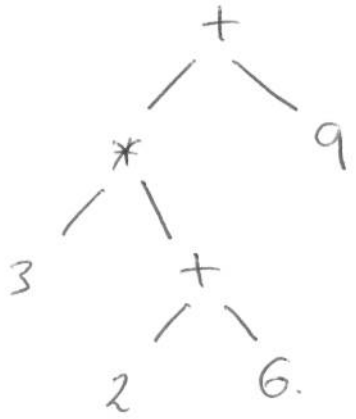
vi)

20 mod 3 = 2
12 mod 3 = 0
25 mod 3 = 1
15 mod 3 = 0.

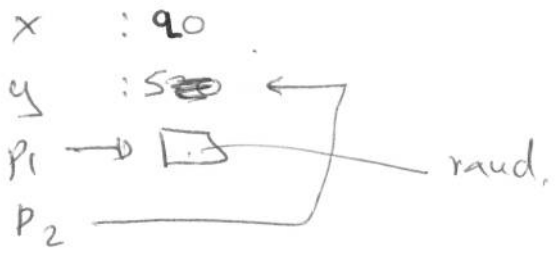
d) i)



ii)



e)



y = *p1 : assigns a random number.

A: y: random num
x = 90

p1 → x
p2 → x

x = 2 · x
y = *p2 = 2 · x

B: x = 40
y = 40

memory leak at p1 = 9x.

f)

i)

```
NodePtr returnPointer(NodePtr hdlist) {
    while (hdlist != NULL) {
        if (hdlist->data >= 10)
            return hdlist;
        hdlist = hdlist->next;
    }
    return hdlist;
}
```

ii)

```
void incOne(NodePtr hdlist) {
    while (hdlist != NULL) {
        hdlist->data = hdlist->data + 1;
        hdlist = hdlist->next;
    }
}
```

2)

a)

```
struct treeNode {
    int id;
    int age;
    int salary;
    treeNode * left;
    treeNode * right;
};

typedef treeNode * treeNodePtr;  (optional)
```

b)

```
void numEmpl. (treeNodePtr hdtree, int &count)

    if (hdtree != NULL) {
        count ++;
        numEmpl (hdtree -> left, count);
        numEmpl (hdtree -> right, count);
    }
}
```

c)

```
void numMang. (treeNodePtr hdtree, int & count) {

    if (hdtree != NULL) {

        if ((hdtree->left == NULL) && (hdtree->right != NULL))
          || (hdtree->left != NULL) && (hdtree->right == NULL)
            count ++;
        numMang (hdtree->left, count);
        numMang (hdtree->right, count);
    }
```

d)

```
void averSalary (treeNodePtr hdtree, int &count, int &salary) {
    if (hdtree != NULL) {
        count ++;
        salary = hdtree->salary / count + salary * (count-1) / count;
        averSalary (hdtree->left, count, salary);
        averSalary (hdtree->right, count, salary);
    }
}
```

e)

```
void peopleUnderID (treeNodePtr, hdtree, int &count, int flag
                    , bool id ) {
    if (hdtree != NULL) {
        if (flag) count ++;
        if (hdtree->id == id)
            flag = true;
        peopleUnderID (hdtree->left, count, flag, id);
        "          hdtree->right, "  "  "
    }
}
```