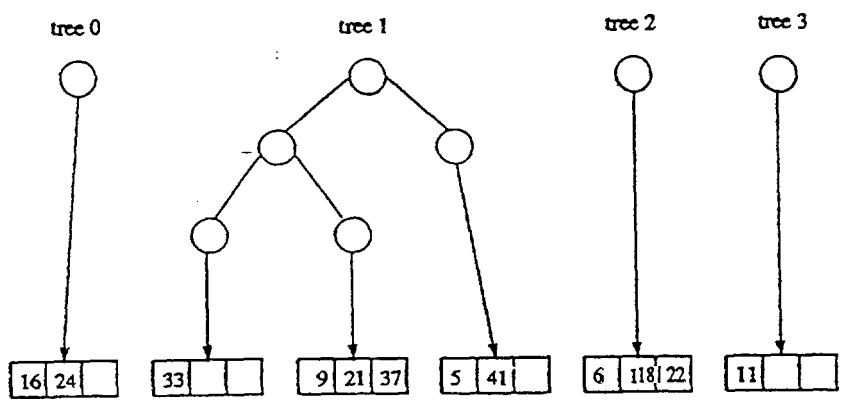
ALGORITHMS FOR DYNAMIC HASHING

```
SEARCH(K) % Find record with key K
    l := root of tree h(K)
    2 := 0
    loop
        exit when l is a leaf of the directory
        if d(K,i) = 0 then l := left(l)
        else l := right(l) end if
        i := i + 1
    end loop
    fetch bucket b pointed by l into main memory
    if b contains record with key K then success
    else failure end if
```

3. (10 points) **The** figure below shows the state of a direct **file** implemented using dynamic hashing. The hash function is $h(K) = K \bmod 4$ and the first few bits of the in&rite bit string associated with each key in the file is given in the table below- Each bucket can hold up to 3 records.

Key	Bit string	Key	Bit string
5	1101.. .	21	0101.. .
6	0110.. .	22	0010.. .
9	0100.. .	24	1100.. .
11	1101.. .	33	0000.. .
16	1011.. .	37	0101.. .
18	1010.. .	41	1000.. .



In the space below draw the state of the **file** after a record with key 17 is inserted, assuming the **first** few bits of the **infinite** bit string associated with 17 are 0100. . . Only those parts of the directory and the data file which have changed as a result of the insertion need to be shown

```
INSERT(R) % Insert record R
    K := key of R
    l := h(K)
    i := 0
    loop
        exit when l is a leaf of the directory
        if d(K, i) = 0 then l := left(l) else l := right(Z) end if
        i := i + 1
    end loop
    fetch bucket b pointed by l into main memory
    if b is not full then
        insert R in b
        write b back into secondary storage
    else % split of b is necessary
        done := false
        loop
            get two new directory nodes l_L and l_R
            set left(l) := l_L and right(l) := l_R
            get a free block b' % b' will become b's buddy
            make l_L point to b and l_R point to b'
            i := i + 1
            for each record R' in b do
                K' := key of R'
                if d(K', i) = 0 then leave R' in b else move R' to b' end if
            end for
            if d(K, i) = 0 then
                if b is not full then
                    insert R into b
                    write b, b' back into secondary storage
                    done := true
                else
                    write b' back into secondary storage
                    l := l_L
                end if
            else % d(K, i) = 1
                if b' is not full then
                    insert R into b'
                    write b, b' back into secondary storage
                    done := true
                else
                    write b back into secondary storage
                    l := l_R
                end if
            end if
            exit when done
        end loop
    end if
```