B-Tree

- B-Tree is also called Balanced & sorted tree.
- It is not a binary tree.
- To reduce disk access, several conditions of the tree must be true.
- The height of the tree must be kept to a minimum
- There must be no empty subtree above the leaves of the tree.
- The leaves of the tree must all be on the same level and all nodes except the leaves must have atleast some minimum number of children.

B Tree of order m has following properties :-

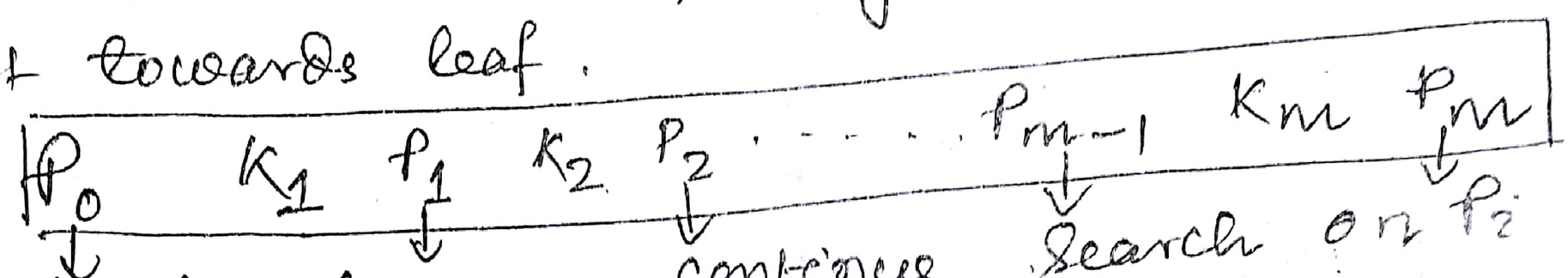
a) It is a m-way search tree in which all leaves are on the same level.

b) All internal nodes except the root will have maximum m children and minimum m/2 children

c) The root node has atmost 'm' children but many have as few as 2 if it is not a leaf.

d) The keys in a node are arranged in the fashion of a search tree.

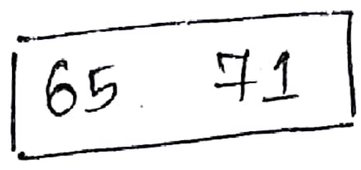
e) The tree is allowed to grow towards the root not towards leaf.



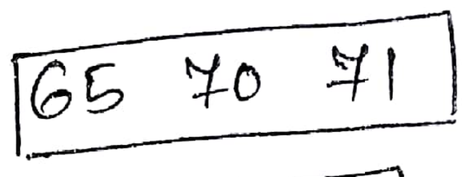
Construction of B tree :-
65, 71, 70, 66, 75, 68, 72, 77, 74, 69, 83, 73, 89, 88, 67, 76, 78, 84, 85, 80 & B tree is of order 5.

Ans i) 65

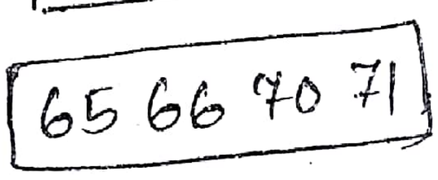
ii) 71



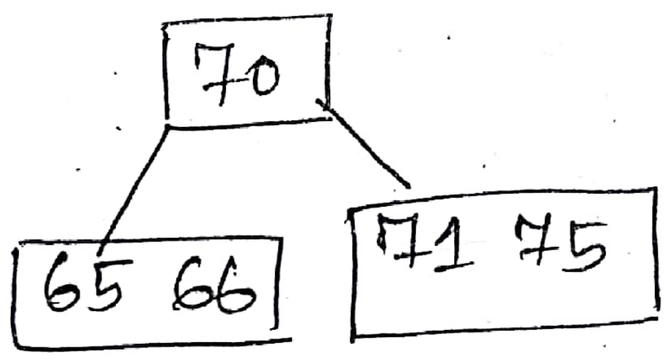
iii) 70



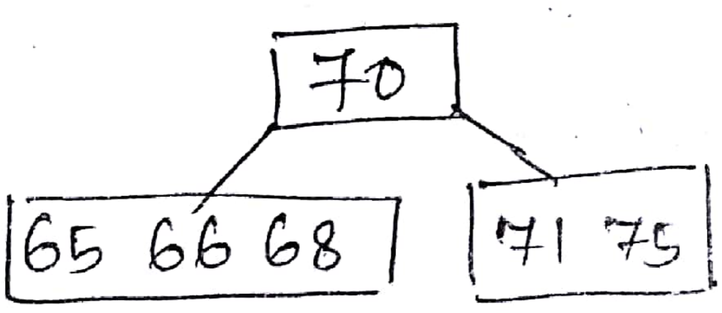
iv) 66



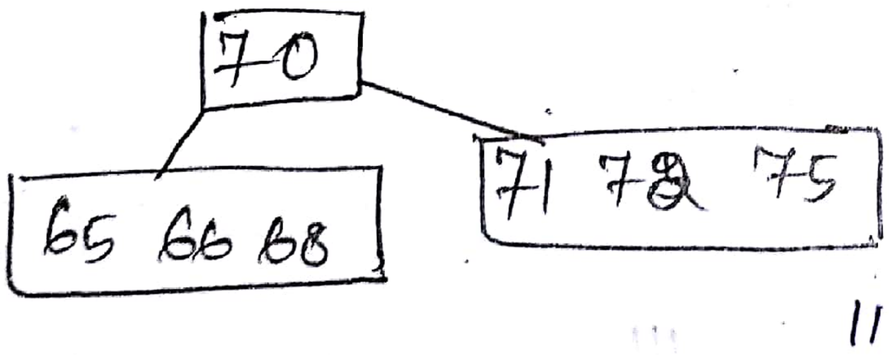
v) 75



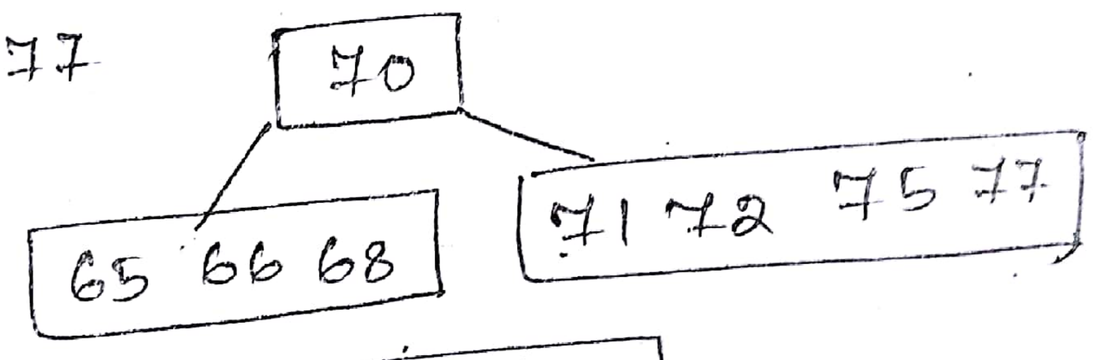
vi) 68



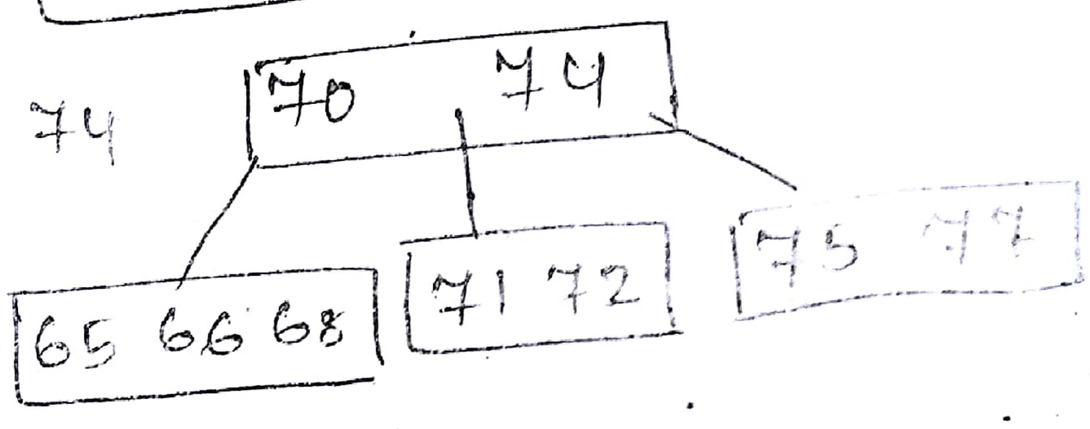
vii) 72



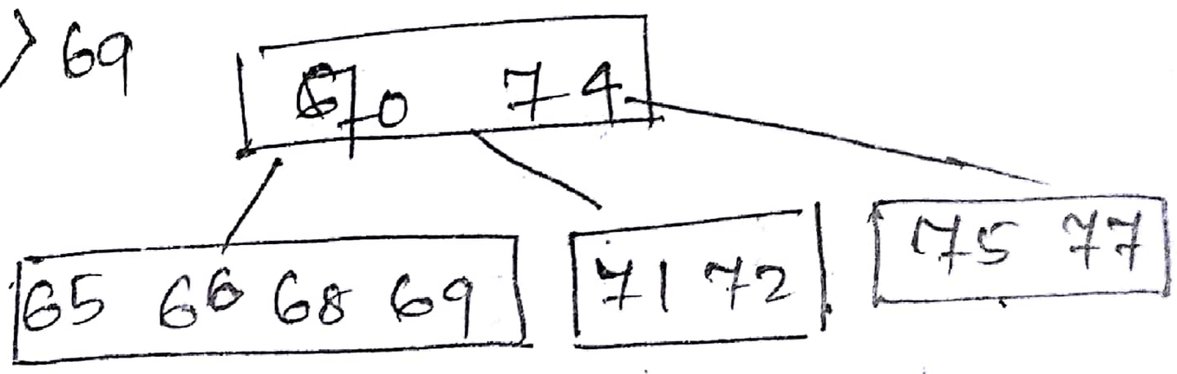
viii) 77



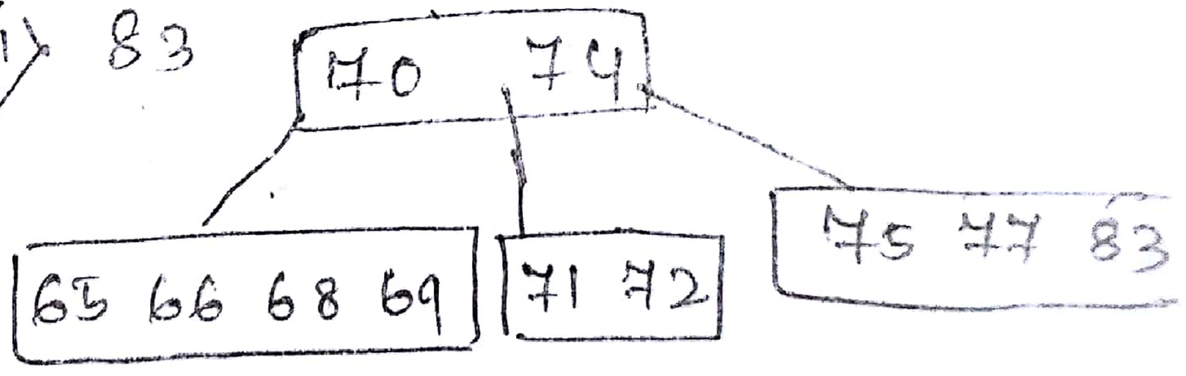
ix) 74



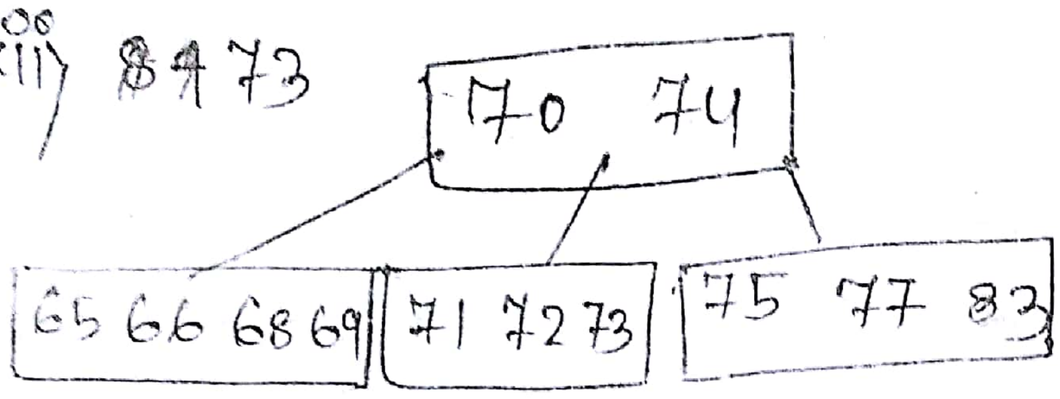
x) 69



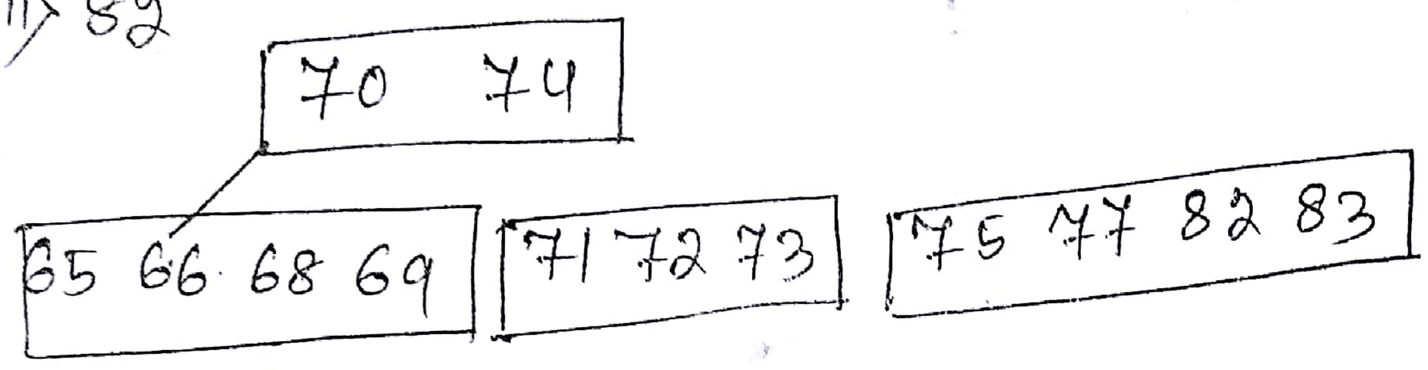
xi) 83



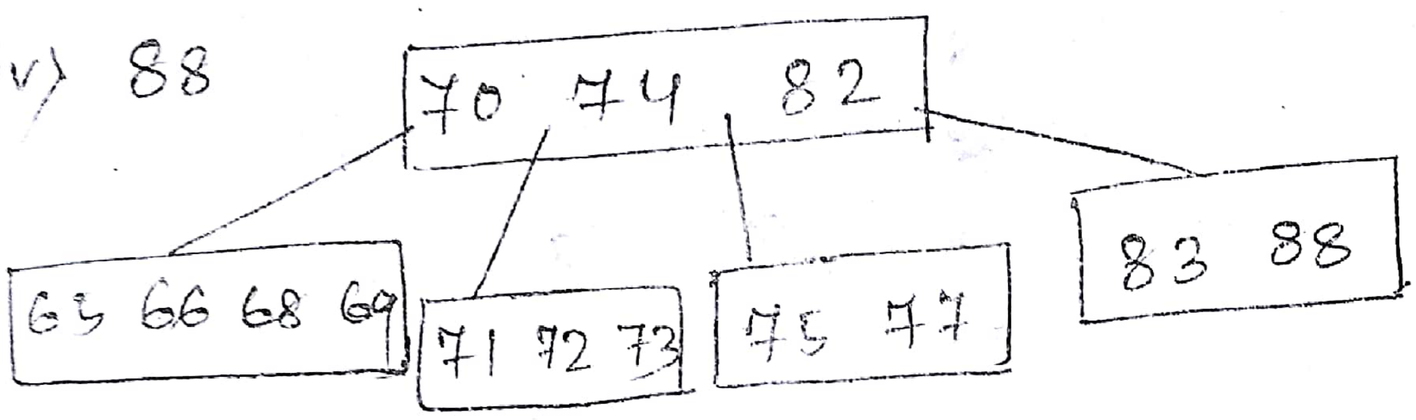
xiii) 8473



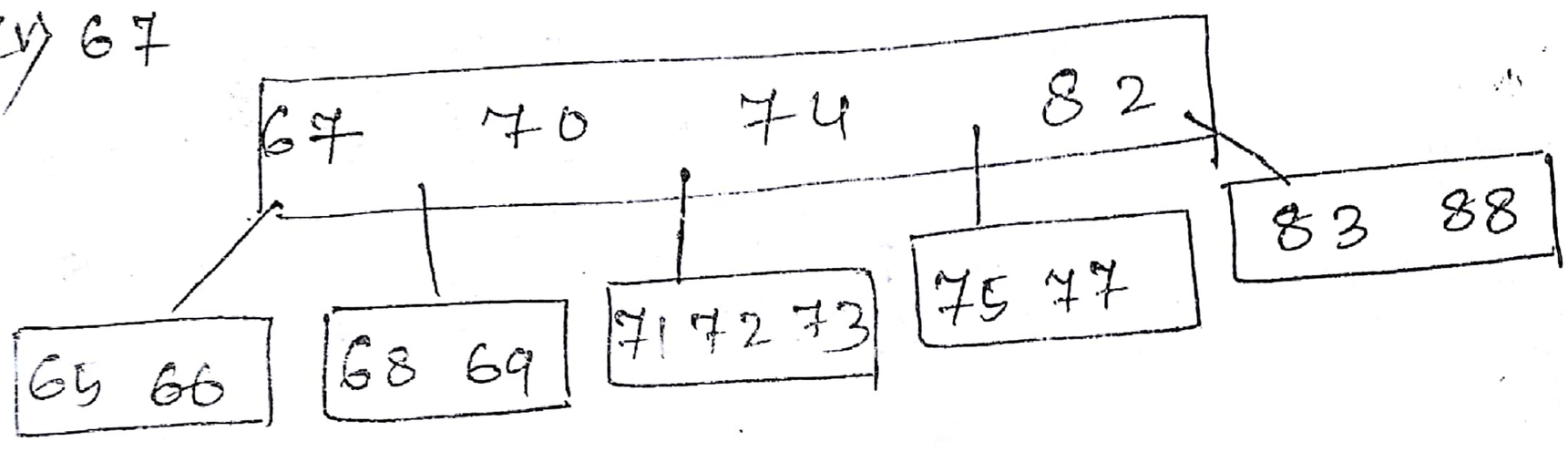
xiii) 82



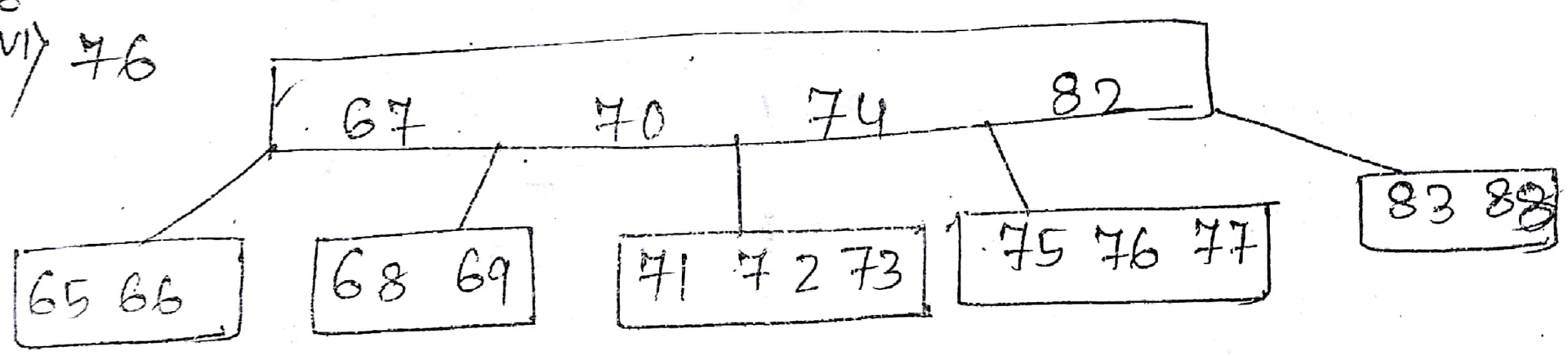
xiv) 88



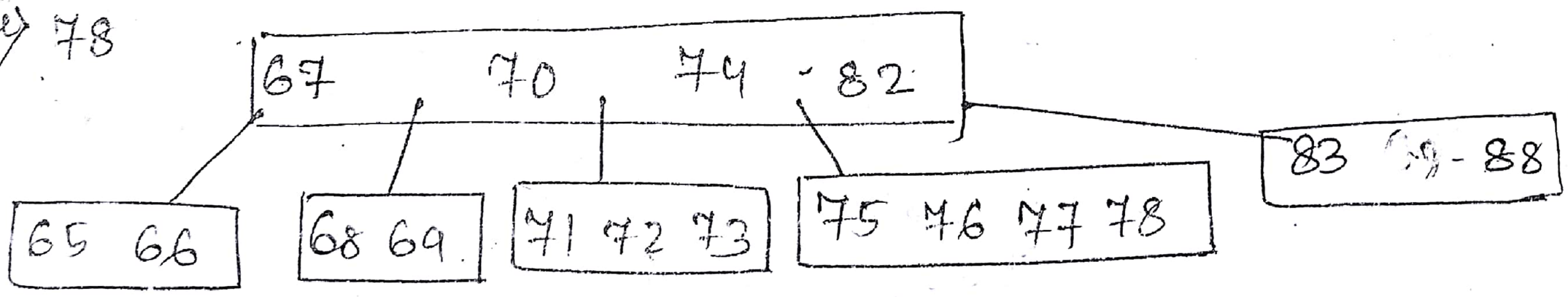
xv) 67



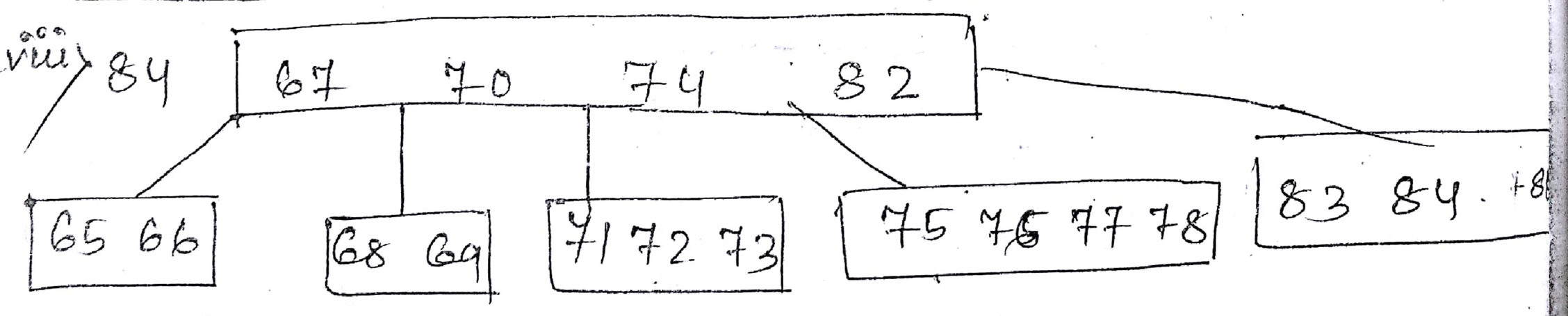
xvi) 76



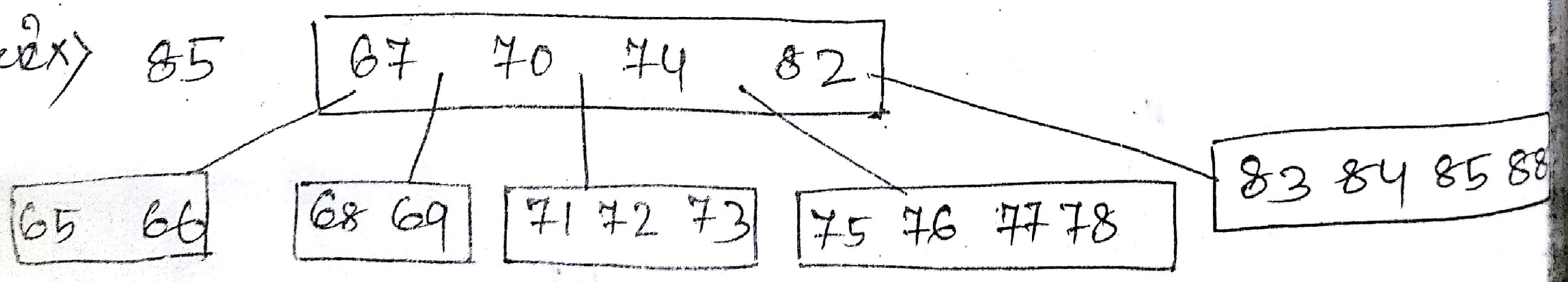
xvii) 78



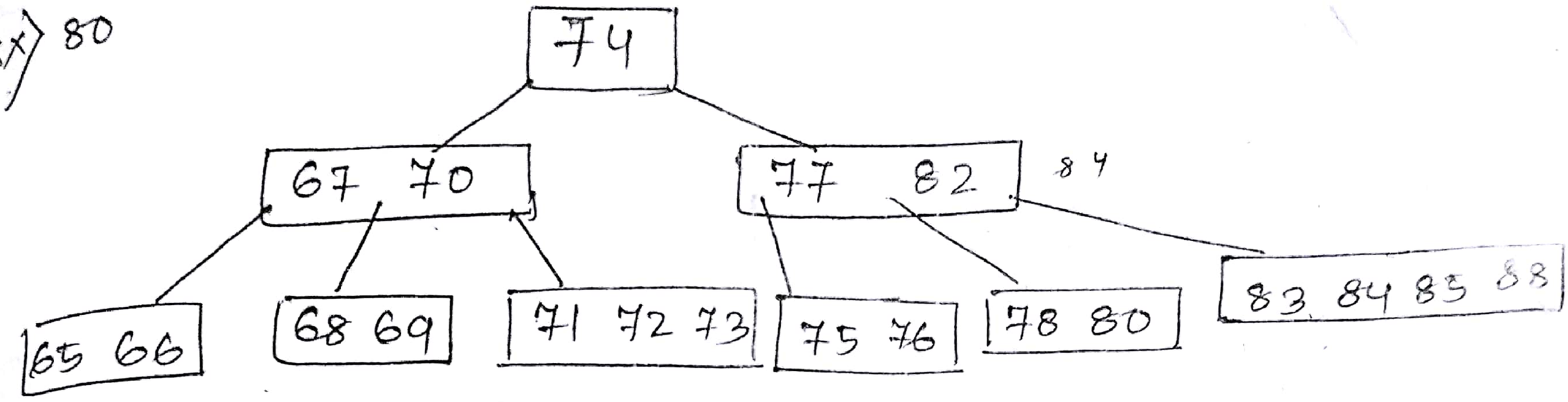
xviii) 84



xix) 85

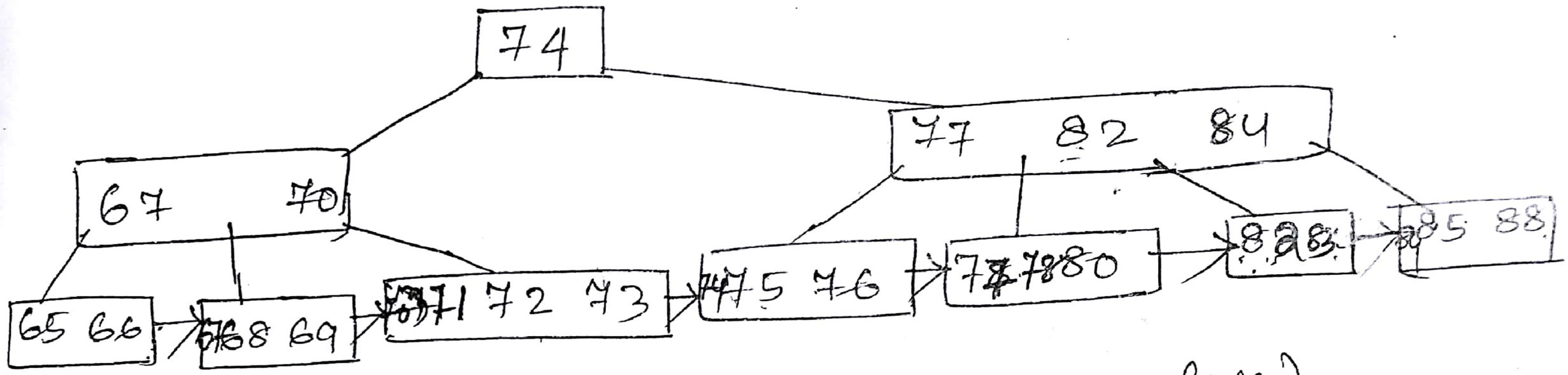


xx) 80



B^+ tree :

In B tree we can access records randomly but sequentially traversal is not possible but in case of B+ tree all the leaf nodes are interconnected i.e a leaf node point to next leaf node



Insertion Algorithm in BST (Binary Search tree)

```
void insert_BST(node *root, int ele)
{node *par, *ptr, *temp;
   temp = root;
   par = NULL;
while( temp != NULL)
 { par = temp;
 if( ele < temp->info)
     temp = temp->lchild;
 else if(ele > temp->info)
     temp = temp->rchild;
 else
 {printf("not found");
    break;
 }
}
ptr = (node *) malloc(sizeof(node));
ptr->info = ele;
ptr->lchild = NULL;
ptr->rchild = NULL;
if(par == NULL)
   par = root;
else if( ele < par->info)
   par->lchild = ptr;
else
   par->rchild = ptr;
}
```