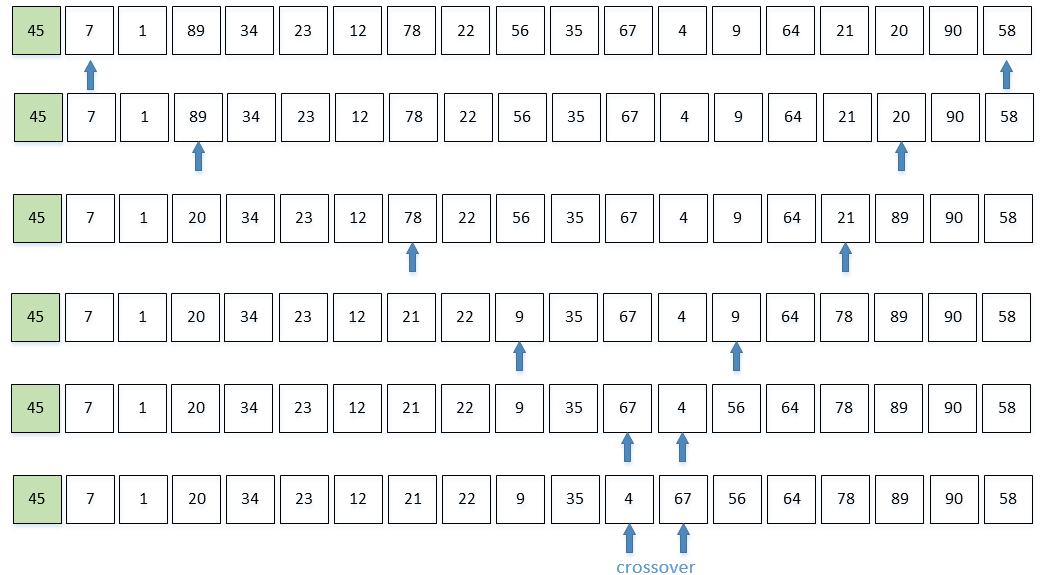
**Assignment 2 (20 marks)**

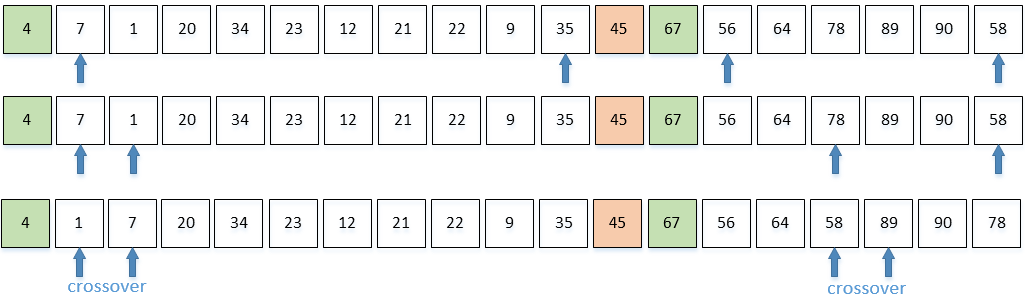
a) Demonstrate how to sort the following data : 45,7,1,89,34,23,12,78,22,56,35,67,4,9,64,21,20,90, 58 using the following sorting techniques

(i) Quicksort

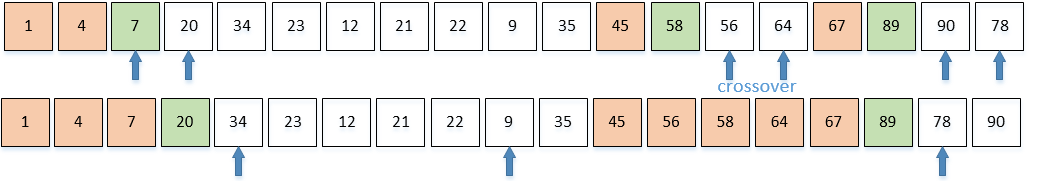
Pass 1



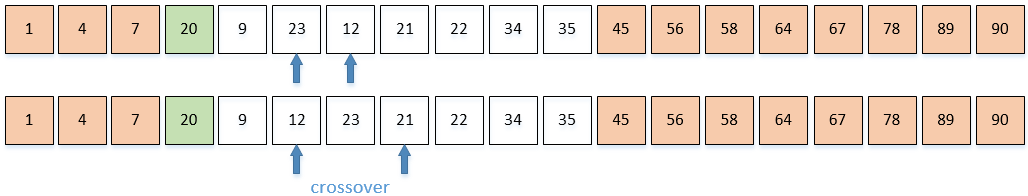
Pass 2



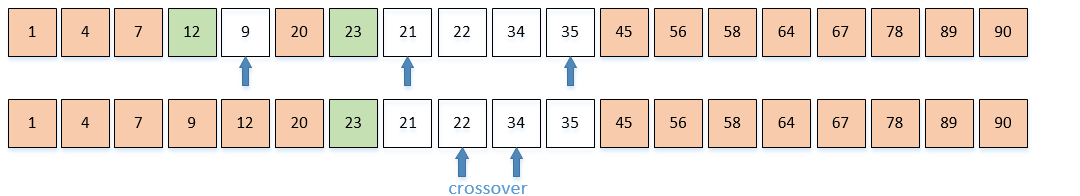
Pass 3



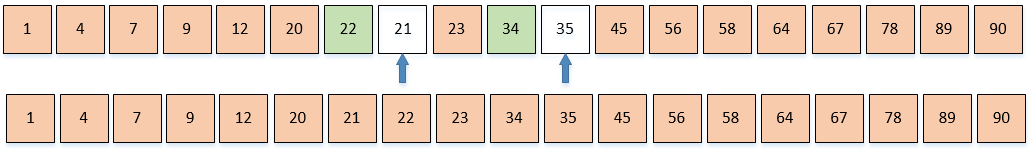
Pass 4



Pass 5



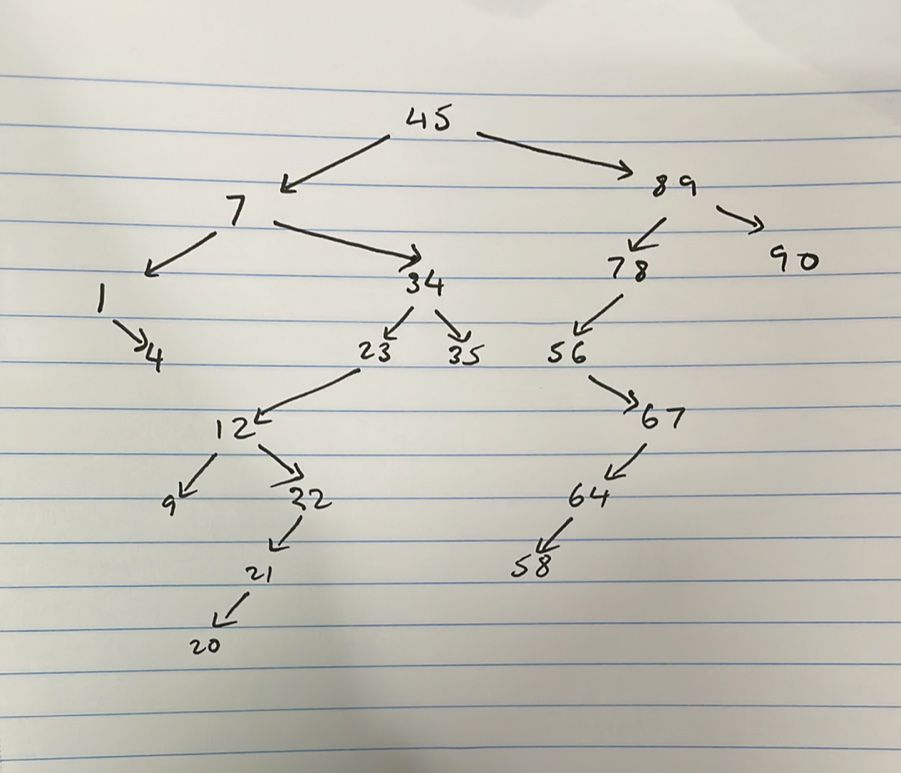
Pass 6



(ii) Mergesort



b) Construct a Binary Search Tree using the data: 45,7,1,89,34,23,12,78,22,56,35,67,4,9,64,21,20,90, 58



c) Using (b) above show the data output after performing **Preorder, Inorder** and **Postorder** traversals

Preorder: 45, 7, 1, 4, 34, 23, 12, 9, 22, 21, 20, 35, 89, 78, 56, 67, 64, 58, 90

Inorder: 1, 4, 7, 9, 12, 20, 21, 22, 23, 34, 35, 45, 56, 58, 64, 67, 78, 89, 90

Postorder: 4, 1, 9, 20, 21, 22, 12, 23, 35, 34, 7, 58, 64, 67, 56, 78, 90, 89, 45

d) Using (b) above write a BST search program for searching the value **21**. The program should **automatically compute** the number of times moves were made to the left and the number of time moves were made to the right in order to locate the value **21**

#include <iostream>

using namespace std;

int countLeft = 0, countRight = 0;

struct node

{

int data;

node \*left, \*right;

};

node \*insert(node \*rootnode, int value)

{

if (rootnode==NULL)

{

node \*newnode = new node;

newnode->data = value;

newnode->left = newnode->right = NULL;

rootnode = newnode;

return rootnode;

}

else if (value<rootnode->data)

{

rootnode->left = insert(rootnode->left, value);

}

else

{

rootnode->right = insert(rootnode->right, value);

}

}

node \*search(node \*rootnode, int value)

{

if (value==rootnode->data)

{

cout<<"found";

}

else if (value<rootnode->data)

{

rootnode->left = search(rootnode->left, value);

countLeft++;

}

else

{

rootnode->right = search(rootnode->right, value);

countRight++;

}

}

int main (){

node \*rootnode = NULL;

int list[19] ={45,7,1,89,34,23,12,78,22,56,35,67,4,9,64,21,20,90,58};

for (int i= 0; i<19; i++)

{

rootnode = insert(rootnode, list[i]);

}

cout << "Searching for value 21"<<endl;

search(rootnode, 21);

cout<<endl;

cout<<"No of left counts: "<<countLeft<< endl<< "No of right counts: "<< countRight;

}

Output:

