Red Black Trees Implementation: -

They are the balanced binary search trees in which each node is colored Red or Black such that

- The root is Black.
- The children of Red Nodes are Black.
- Each path contains same number of Black Nodes.
- Leaves of this tree are null nodes and each of these leave nodes are Black.

Implementation: -

I have implemented Red-Black Tree data-structure for storing processes with following Functions:

1)tr_insert():-

This function is used for inserting new processes into the tree just in the manner, values are inserted in a simple binary search tree. Just that here the nodes will be colored. The insertion may violate the node coloring rules and so for fixing it up, I have implemented another function which rotates the nodes satisfying the rules and fixes up the insertion.

2)tr_print():-

This function is used for printing the values which have been inserted in the tree.

3)tr_TreeCreate():-

This function basically allocates and initializes memory for the tree.

4) tr_search():-

This function is created to search the particular value from the tree and I have also called delete function inside this function to delete the value which is being searched for.

5)callprocess():-

I have created this function to call the processes and inserting them to the tree and then after suspending it.

Test Cases: -

For testing the working of my Red Black Tree Implementation, I have created four processes for "callprocess" function. So this is inserting these four processes to the tree and then after to test the working of search functionality, I am checking when there are more than 2 processes inserted in the tree then my code should search for the previous value which has been inserted in the tree. Then after print the entire tree nodes at the end.

The following is the screenshot which depicts this –