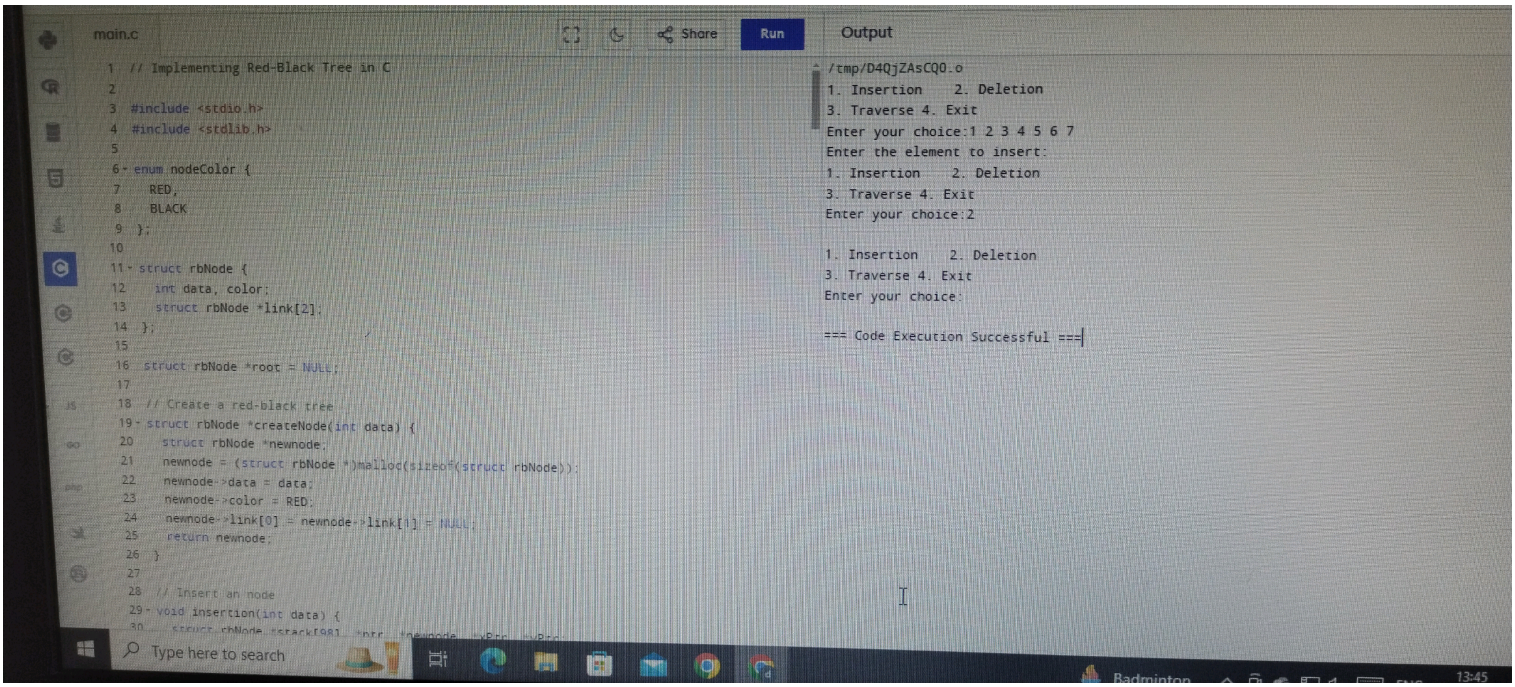

main.c
Share
Run
Output
// Implementing Red-Black Tree in C
#include <stdio.h>
#include <stdlib.h>
enum nodeColor {
RED,
BLACK
};
struct rbNode {
int data, color;
struct rbNode *link[2];
};
struct rbNode *root = NULL;
// Create a red-black tree
struct rbNode *createNode(int data) {
struct rbNode *newnode;
newnode = (struct rbNode *)malloc(sizeof(struct rbNode));
newnode->data = data;
newnode->color = RED;
newnode->link[0] = newnode->link[1] = NULL;
return newnode;
}
// Insert an node
/tmp/D4QjZAsCQO.o
1. Insertion 2. Deletion
3. Traverse 4. Exit
Enter your choice:1 2 3 4 5 6 7
Enter the element to insert:
1. Insertion 2. Deletion
3. Traverse 4. Exit
Enter your choice:2
1. Insertion 2. Deletion
3. Traverse 4. Exit
Enter your choice:
=== Code Execution Successful ===
Type here to search
Badminton
13:45

main.c

Share Run

```
int dir[98], ht = 0, index;
ptr = root;
if (!root) {
  root = createNode(data);
  return;
}

stack[ht] = root;
dir[ht++] = 0;
while (ptr != NULL) {
  if (ptr->data == data) {
    printf("Duplicates Not Allowed!!\n");
    return;
  }
  index = (data - ptr->data) > 0 ? 1 : 0;
  stack[ht] = ptr;
  ptr = ptr->link[index];
  dir[ht++] = index;
}
stack[ht - 1]->link[index] = newnode = createNode(data);
while ((ht >= 3) && (stack[ht - 1]->color == RED)) {
  if (dir[ht - 2] == 0) {
    yPtr = stack[ht - 2]->link[1];
    if (yPtr != NULL && yPtr->color == RED) {
      stack[ht - 2]->color = RED;
      stack[ht - 1]->color = yPtr->color = BLACK;
      ht = ht - 2;
    } else {
      if (dir[ht - 1] == 0) {
        yPtr = stack[ht - 1];
```

Output

```
/tmp/D4QjZAsCQ0.o
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:1 2 3 4 5 6 7
Enter the element to insert:
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:2

1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:

=== Code Execution Successful ===
```

main.c

Share Run

```
62          xPtr = stack[ht - 1];
63          yPtr = xPtr->link[1];
64          xPtr->link[1] = yPtr->link[0];
65          yPtr->link[0] = xPtr;
66          stack[ht - 2]->link[0] = yPtr;
67        }
68        xPtr = stack[ht - 2];
69        xPtr->color = RED;
70        yPtr->color = BLACK;
71        xPtr->link[0] = yPtr->link[1];
72        yPtr->link[1] = xPtr;
73        if (xPtr == root) {
74          root = yPtr;
75        } else {
76          stack[ht - 3]->link[dir[ht - 3]] = yPtr;
77        }
78        break;
79      }
80    } else {
81      yPtr = stack[ht - 2]->link[0];
82      if ((yPtr != NULL) && (yPtr->color == RED)) {
83        stack[ht - 2]->color = RED;
84        stack[ht - 1]->color = yPtr->color = BLACK;
85        ht = ht - 2;
86      } else {
87        if (dir[ht - 1] == 1) {
88          yPtr = stack[ht - 1];
89        } else {
90          xPtr = stack[ht - 1];
91          yPtr = xPtr->link[0];
```

Output

```
/tmp/D4QjZAsCQ0.o
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:1 2 3 4 5 6 7
Enter the element to insert:
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:2

1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:

=== Code Execution Successful ===
```

Type here to search

main.c

Share | Run | Output

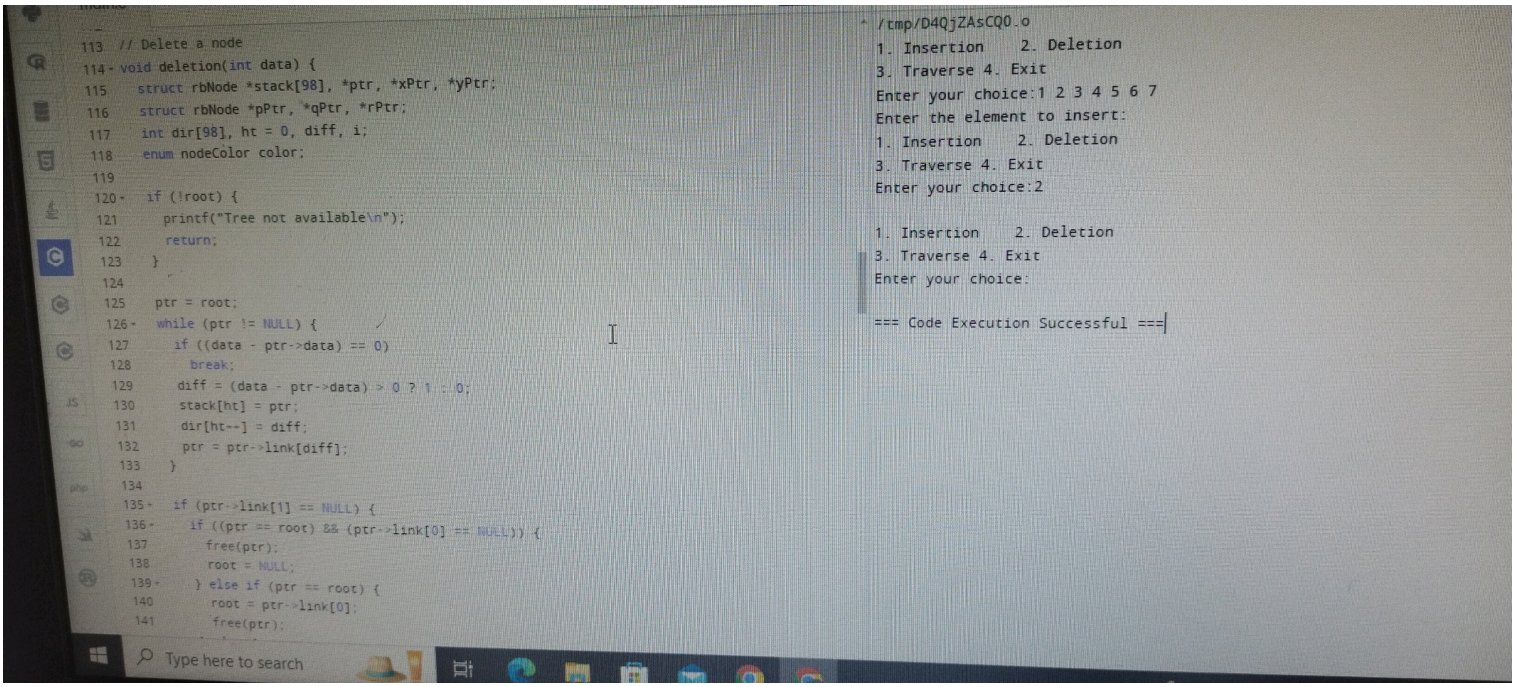
```
    } else {
      if (dir[ht - 1] == 1) {
        yPtr = stack[ht - 1];
      } else {
        xPtr = stack[ht - 1];
        yPtr = xPtr->link[0];
        xPtr->link[0] = yPtr->link[1];
        yPtr->link[1] = xPtr;
        stack[ht - 2]->link[1] = yPtr;
      }
      xPtr = stack[ht - 2];
      yPtr->color = BLACK;
      xPtr->color = RED;
      xPtr->link[1] = yPtr->link[0];
      yPtr->link[0] = xPtr;
      if (xPtr == root) {
        root = yPtr;
      } else {
        stack[ht - 3]->link[dir[ht - 3]] = yPtr;
      }
      break;
    }
  }
  root->color = BLACK;
}

// Delete a node
```

```
/tmp/D4QjZAsCQO.o
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:1 2 3 4 5 6 7
Enter the element to insert:
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:2

1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:

=== Code Execution Successful ===
```



```
// Delete a node
void deletion(int data) {
  struct rbNode *stack[98], *ptr, *xPtr, *yPtr;
  struct rbNode *pPtr, *qPtr, *rPtr;
  int dir[98], ht = 0, diff, i;
  enum nodeColor color;

  if (!root) {
    printf("Tree not available\n");
    return;
  }

  ptr = root;
  while (ptr != NULL) {
    if ((data - ptr->data) == 0)
      break;
    diff = (data - ptr->data) > 0 ? 1 : 0;
    stack[ht] = ptr;
    dir[ht++] = diff;
    ptr = ptr->link[diff];
  }

  if (ptr->link[1] == NULL) {
    if ((ptr == root) && (ptr->link[0] == NULL)) {
      free(ptr);
      root = NULL;
    } else if (ptr == root) {
      root = ptr->link[0];
      free(ptr);
```

```
/tmp/D4QjZAsCQO.o
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:1 2 3 4 5 6 7
Enter the element to insert:
1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:2

1. Insertion    2. Deletion
3. Traverse 4. Exit
Enter your choice:

=== Code Execution Successful ===
```

Type here to search