**LAB ASSIGNMENT**

**WEEK (20-25 April)**

**Q1. WAP for a Binary Search Tree that returns the maximum and minimum values in it. (using recursion)**

***Code:-***

*#include<stdio.h>*

*#include<stdlib.h>*

*struct node{*

*int data;*

*struct node \*left,\*right;*

*};*

*void append(struct node\*\*q , int n)*

*{*

*if(\*q == NULL)*

*{*

*(\*q)=malloc(sizeof(struct node));*

*(\*q)->data = n;*

*(\*q)->left = NULL;*

*(\*q)->right =NULL;*

*}*

*else*

*{*

*if(n<(\*q)->data)*

*append(&((\*q)->left),n);*

*else*

*append(&((\*q)->right),n);*

*}*

*}*

*int min(struct node \*\*q)*

*{*

*struct node \*p;*

*p=\*q;*

*if(p->left==NULL)*

*return p->data;*

*else*

*return min(&(p->left));*

*}*

*int max(struct node \*\*q)*

*{*

*struct node \*p;*

*p=\*q;*

*if(p->right==NULL)*

*return p->data;*

*else*

*return max(&(p->right));*

*}*

*int main()*

*{*

*int ch,n;*

*struct node \*p;*

*p=NULL;*

*printf("Choice:-\n");*

*printf("1. Insertion\n");*

*printf("2. Minimum Value\n");*

*printf("3. Maximum value\n");*

*printf("6. Exit\n\n");*

*while(1)*

*{*

*printf("Enter your choice:");*

*scanf("%d",&ch);*

*switch(ch)*

*{*

*case 1:printf("Enter data:");*

*scanf("%d",&n);*

*append(&p,n);*

*break;*

*case 2:if(p==NULL)*

*printf("\nEmpty Tree\n\n");*

*else*

*printf("\nMinimum Value : %d\n\n",min(&p));*

*break;*

*case 3:if(p==NULL)*

*printf("\nEmpty Tree\n\n");*

*else*

*printf("\nMaximum Value : %d\n\n",max(&p));*

*break;*

*default:*

*break;*

*}*

*if(ch==4)*

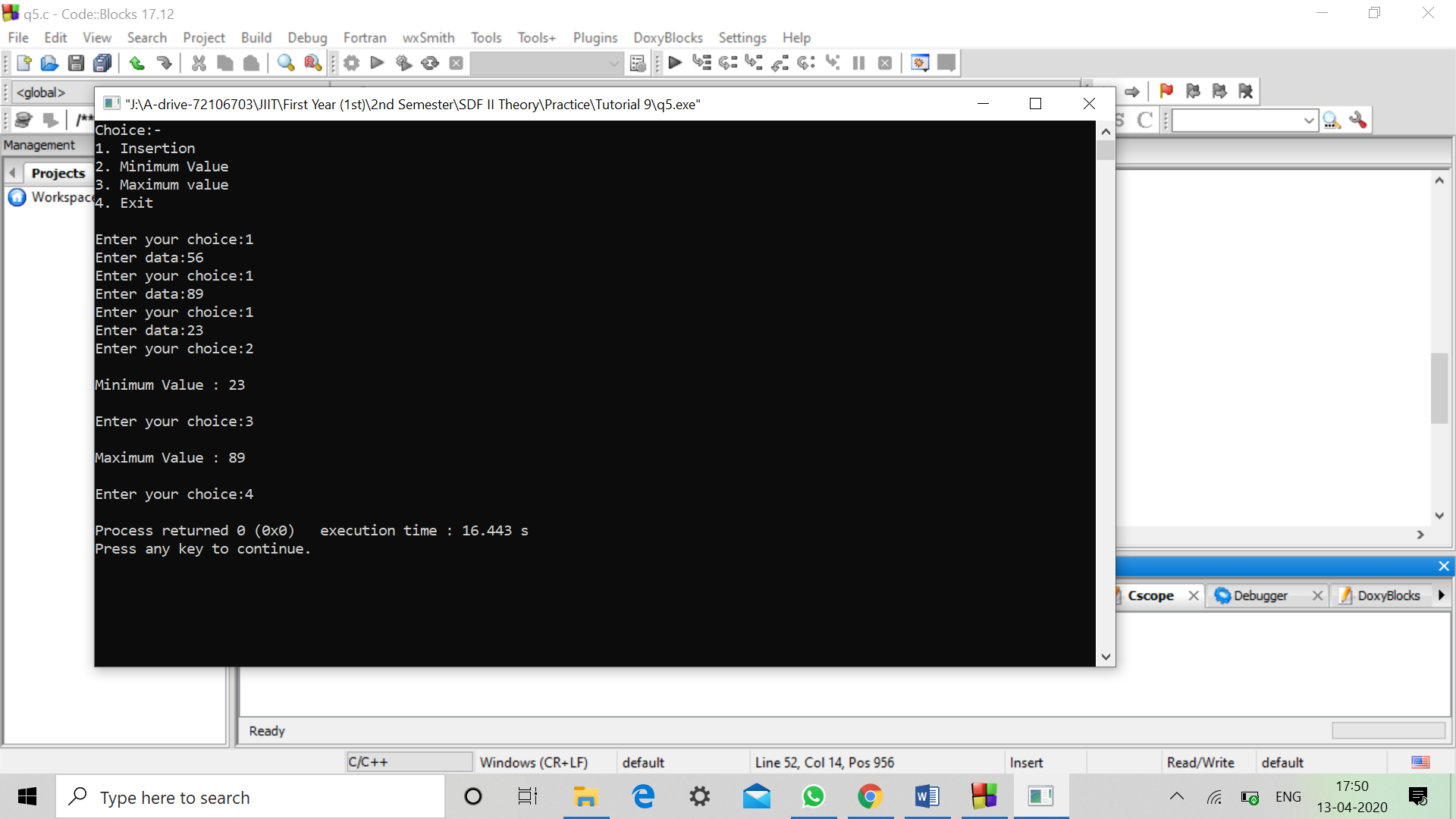
*break;*

*}*

*return 0;*

*}*

***Output:-***

******

**Q2. Implement the following functions on Binary Trees:**

**a) Count the number of nodes**

**b) Count the number of leaves**

**c) Count the number of right children**

**d) Find the height of the tree**

***Code:-***

*#include<stdio.h>*

*#include<stdlib.h>*

*struct node{*

*int data;*

*struct node \*left,\*right;*

*};*

*void append(struct node\*\*q , int n)*

*{*

*if(\*q == NULL)*

*{*

*(\*q)=malloc(sizeof(struct node));*

*(\*q)->data = n;*

*(\*q)->left = NULL;*

*(\*q)->right =NULL;*

*}*

*else*

*{*

*if(n<(\*q)->data)*

*append(&((\*q)->left),n);*

*else*

*append(&((\*q)->right),n);*

*}*

*}*

*int tnode(struct node \*\*q)*

*{*

*struct node \*p;*

*p=\*q;*

*if(p!=NULL)*

*return (tnode(&(p->right))+tnode(&(p->left))+1);*

*else*

*return 0;*

*}*

*int tleaves(struct node \*\*q)*

*{*

*struct node \*p;*

*p=\*q;*

*if(p->left==NULL && p->right==NULL)*

*return 1;*

*else if(p->left==NULL && p->right!=NULL)*

*return tleaves(&(p->right));*

*else if(p->left!=NULL && p->right==NULL)*

*return tleaves(&(p->left));*

*else*

*return (tleaves(&(p->right))+tleaves(&(p->left)));*

*}*

*int x=0;*

*int trch(struct node \*\*q)*

*{*

*struct node \*p;*

*p=\*q;*

*if(p->left!=NULL)*

*trch(&(p->left));*

*if(p->right!=NULL)*

*{*

*x++;*

*trch(&(p->right));*

*}*

*return x;*

*}*

*int height(struct node \*\*q)*

*{*

*int h1,h2;*

*struct node \*p;*

*if(\*q==NULL)*

*return 0;*

*else*

*{*

*p=\*q;*

*h1=height(&(p->left));*

*h2=height(&(p->right));*

*if(h1<=h2)*

*return(h2+1);*

*else*

*return(h1+1);*

*}*

*}*

*int main()*

*{*

*int ch,n;*

*struct node \*p;*

*p=NULL;*

*printf("Choice:-\n");*

*printf("1. Insertion\n");*

*printf("2. Total Nodes\n");*

*printf("3. Total leaves\n");*

*printf("4. Total right childrens\n");*

*printf("5. Height\n");*

*printf("6. Exit\n\n");*

*while(1)*

*{*

*printf("Enter your choice:");*

*scanf("%d",&ch);*

*switch(ch)*

*{*

*case 1:printf("Enter data:");*

*scanf("%d",&n);*

*append(&p,n);*

*break;*

*case 2:if(p==NULL)*

*printf("\nEmpty Tree\n\n");*

*else*

*printf("\nTotal no. of Nodes : %d\n\n",tnode(&p));*

*break;*

*case 3:if(p==NULL)*

*printf("\nEmpty Tree\n\n");*

*else*

*printf("\nTotal no. of Leaves : %d\n\n",tleaves(&p));*

*break;*

*case 4: x=0;*

*if(p==NULL)*

*printf("\nEmpty Tree\n\n");*

*else*

*printf("\nTotal no. of right childrens : %d\n\n",trch(&p));*

*break;*

*case 5:if(p==NULL)*

*printf("\nEmpty Tree\n\n");*

*else*

*printf("\nHeight : %d\n\n",height(&p));*

*break;*

*default:*

*break;*

*}*

*if(ch==6)*

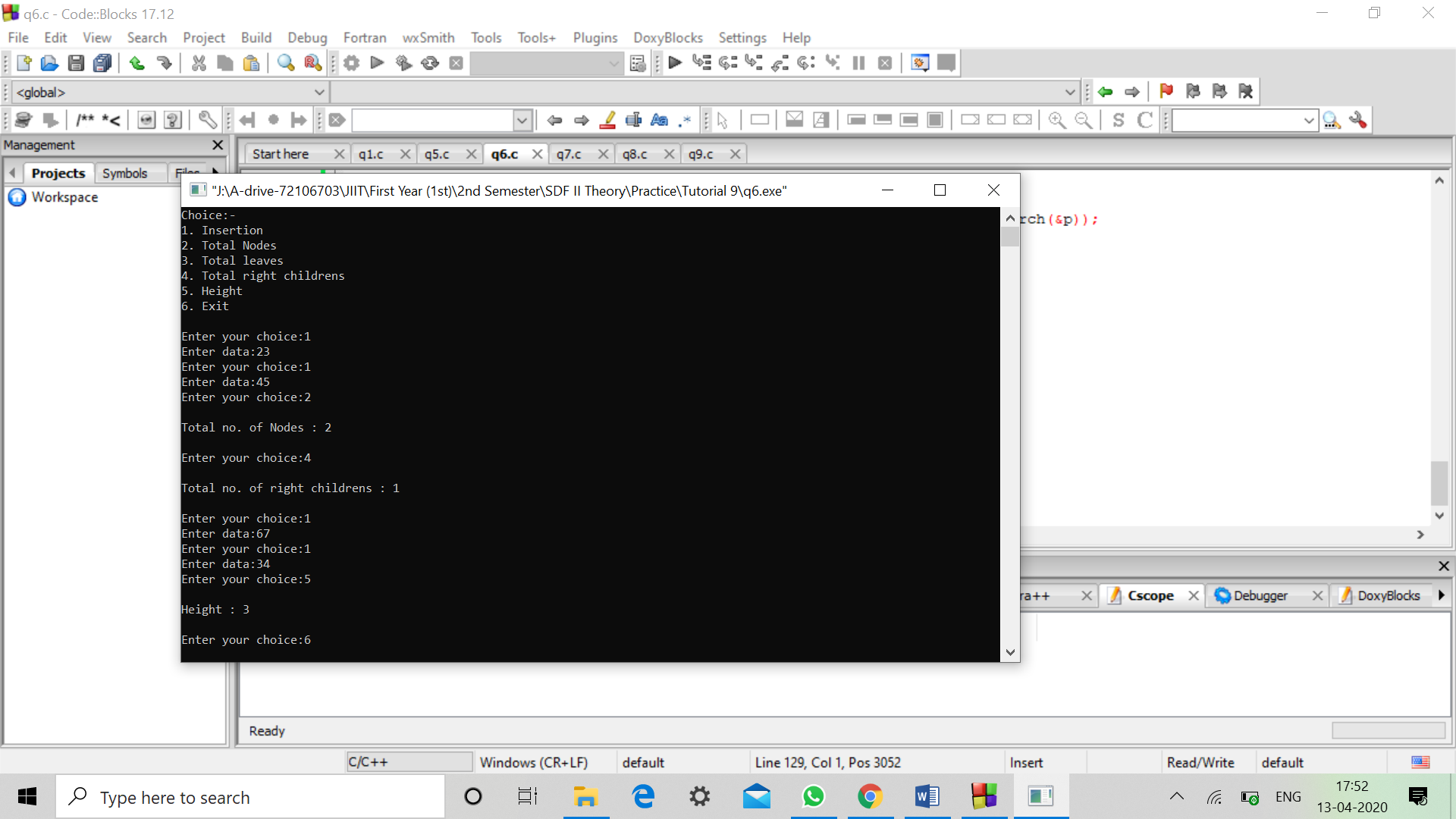
*break;*

*}*

*return 0;*

*}*

***Output:-***



**Q3. WAP to check if all the leaves of a Binary Tree are at same level or not.**

***Code:-***

*#include<stdio.h>*

*#include<stdlib.h>*

*struct node{*

*int data;*

*struct node \*left,\*right;*

*};*

*void append(struct node\*\*q , int n)*

*{*

*if(\*q == NULL)*

*{*

*(\*q)=malloc(sizeof(struct node));*

*(\*q)->data = n;*

*(\*q)->left = NULL;*

*(\*q)->right =NULL;*

*}*

*else*

*{*

*if(n<(\*q)->data)*

*append(&((\*q)->left),n);*

*else*

*append(&((\*q)->right),n);*

*}*

*}*

*int f=1;*

*void check(struct node \*\*q,int n,int h)*

*{*

*struct node \*p;*

*p=\*q;*

*if(n==1)*

*if(p->left==NULL || p->right==NULL)*

*if(n!=h)*

*f=0;*

*if(p->left==NULL && p->right==NULL)*

*if(n!=h)*

*f=0;*

*if(p->right!=NULL)*

*check(&(p->right),n+1,h);*

*if(p->left!=NULL)*

*check(&(p->left),n+1,h);*

*}*

*int height(struct node \*\*q)*

*{*

*int h1,h2;*

*struct node \*p;*

*if(\*q==NULL)*

*return 0;*

*else*

*{*

*p=\*q;*

*h1=height(&(p->left));*

*h2=height(&(p->right));*

*if(h1<=h2)*

*return(h2+1);*

*else*

*return(h1+1);*

*}*

*}*

*int main()*

*{*

*int ch,n,h;*

*struct node \*p;*

*p=NULL;*

*printf("Choice:-\n");*

*printf("1. Insertion\n");*

*printf("2. Check whether all leaves are at same level or not \n");*

*printf("3. Exit\n\n");*

*while(1)*

*{*

*printf("Enter your choice:");*

*scanf("%d",&ch);*

*switch(ch)*

*{*

*case 1:printf("Enter data:");*

*scanf("%d",&n);*

*append(&p,n);*

*break;*

*case 2: f=1;*

*h=height(&p);*

*check(&p,1,h);*

*if(f==0)*

*printf("All leaves are not at same level.\n");*

*else*

*printf("All leaves are at same level.\n");*

*break;*

*default:*

*break;*

*}*

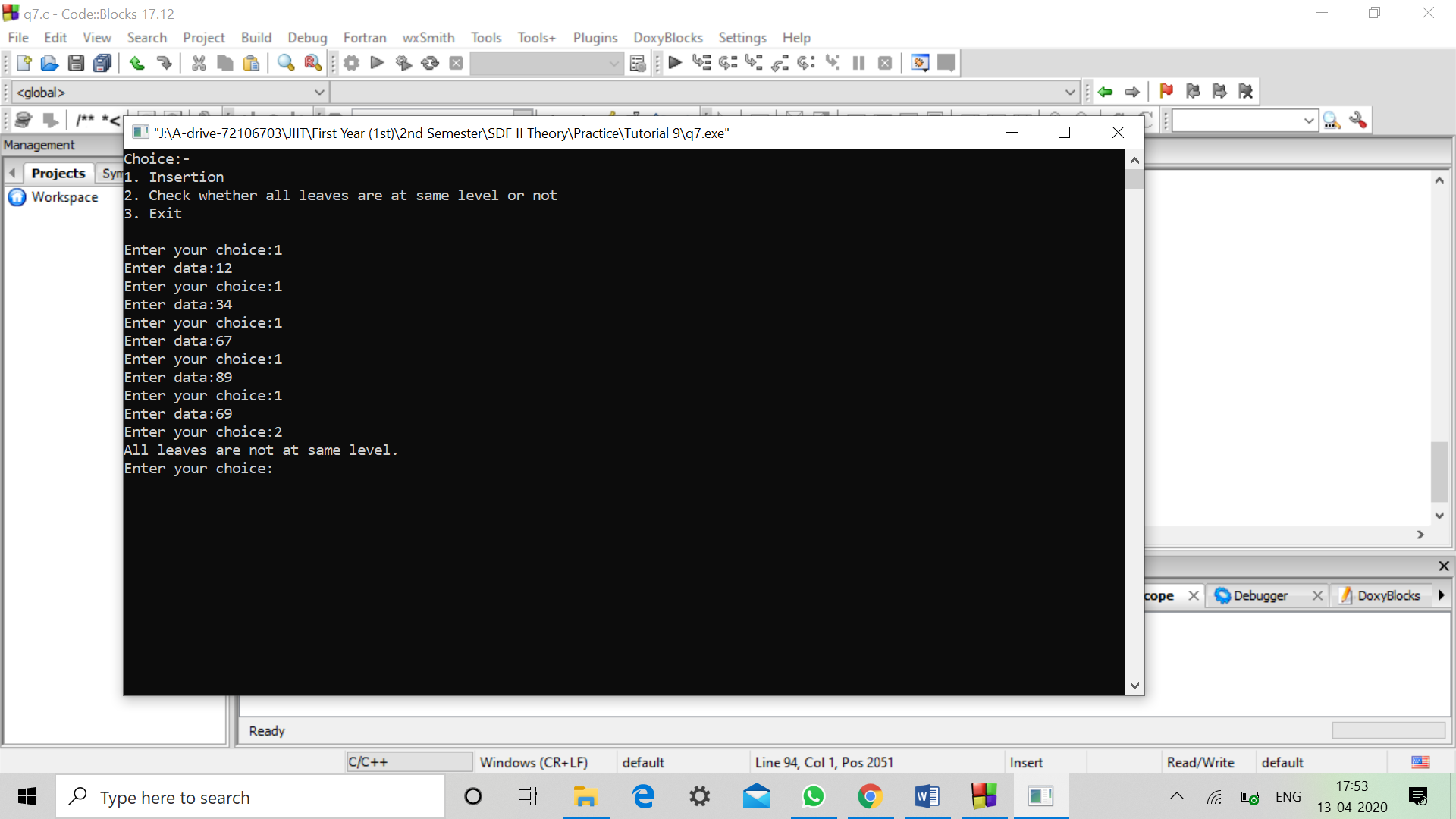
*if(ch==3)*

*break; }*

*return 0;*

*}*

***Output:-***

******

**Q4. WAP to check if a Binary Tree has duplicate values or not.**

***Code:-***

*#include<stdio.h>*

*#include<stdlib.h>*

*struct node{*

*int data;*

*struct node \*left,\*right;*

*};*

*void append(struct node\*\*q , int n)*

*{*

*if(\*q == NULL)*

*{*

*(\*q)=malloc(sizeof(struct node));*

*(\*q)->data = n;*

*(\*q)->left = NULL;*

*(\*q)->right =NULL;*

*}*

*else*

*{*

*if(n==(\*q)->data)*

*append(&((\*q)->right),n);*

*else*

*append(&((\*q)->left),n);*

*}*

*}*

*int x=0;*

*int check(struct node \*\*q)*

*{*

*struct node \*p;*

*p=\*q;*

*if(p->left!=NULL)*

*check(&(p->left));*

*if(p->right!=NULL)*

*{*

*x++;*

*check(&(p->right));*

*}*

*return x;*

*}*

*int main()*

*{*

*int ch,n,h;*

*struct node \*p;*

*p=NULL;*

*printf("Choice:-\n");*

*printf("1. Insertion\n");*

*printf("2. Check whether Binary Tree (not BST) has duplicate values \n");*

*printf("3. Exit\n\n");*

*while(1)*

*{*

*printf("Enter your choice:");*

*scanf("%d",&ch);*

*switch(ch)*

*{*

*case 1:printf("Enter data:");*

*scanf("%d",&n);*

*append(&p,n);*

*break;*

*case 2: x=0;*

*printf("Binary Tree (not BST) has %d duplicate values.\n",check(&p));*

*break;*

*default:*

*break;*

*}*

*if(ch==3)*

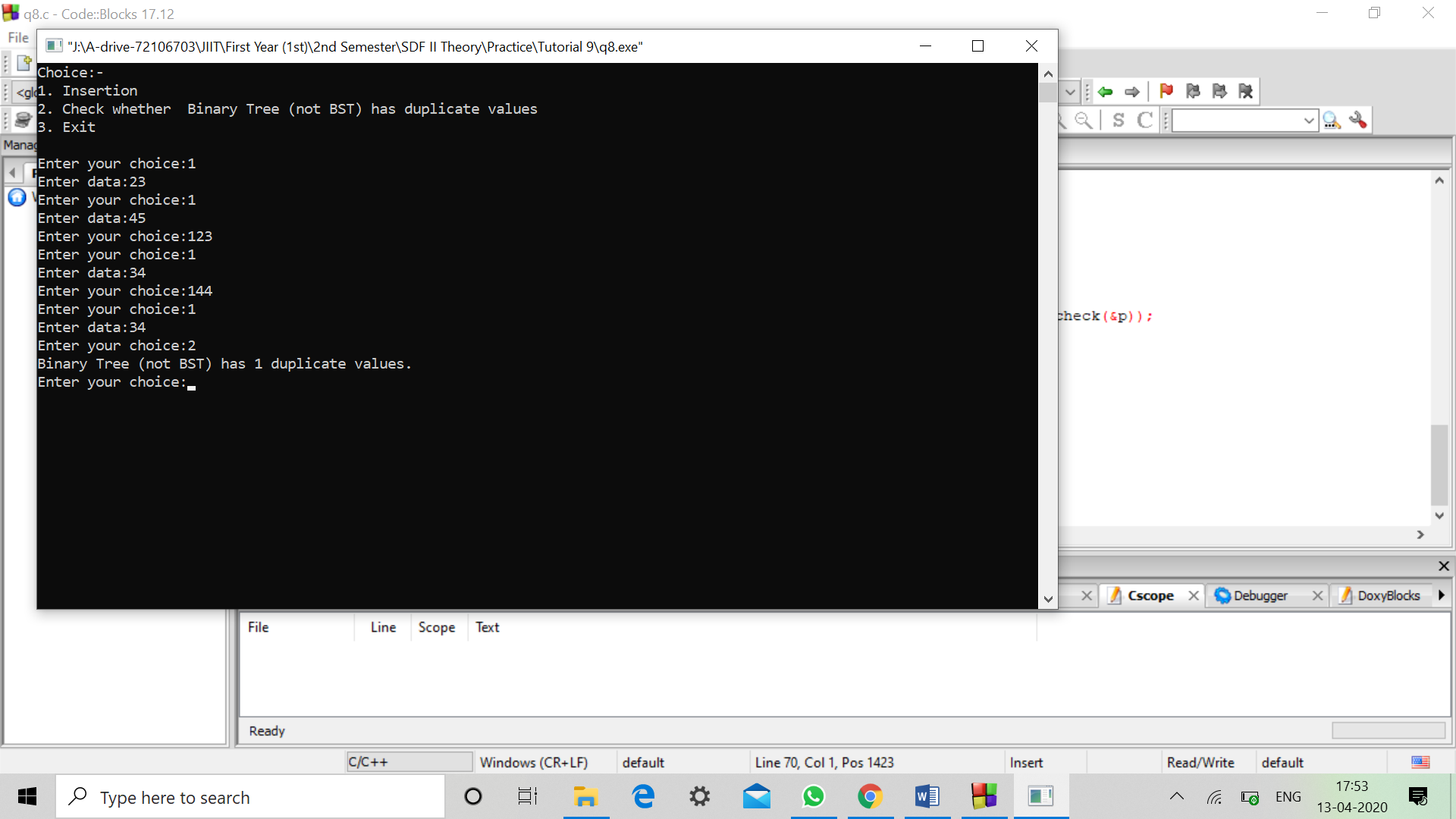
*break;*

*}*

*return 0;*

*}*

***Output:-***

******