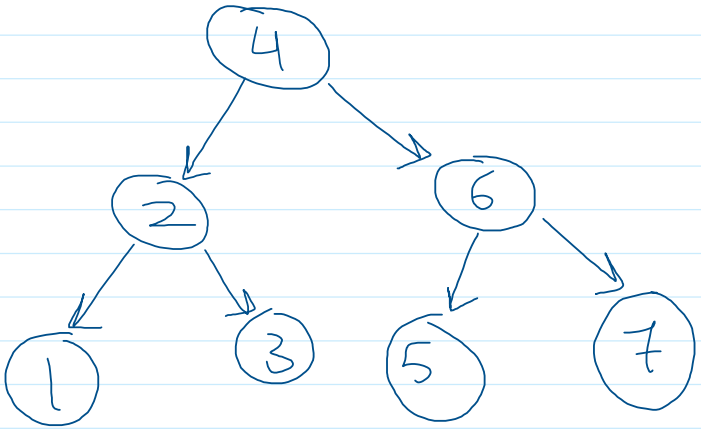
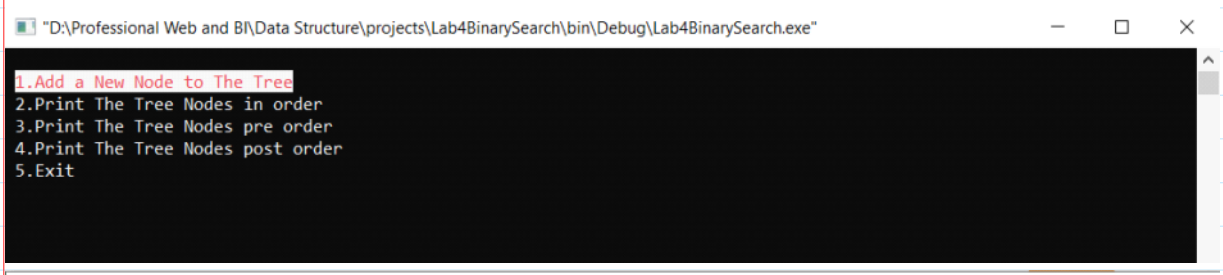
1

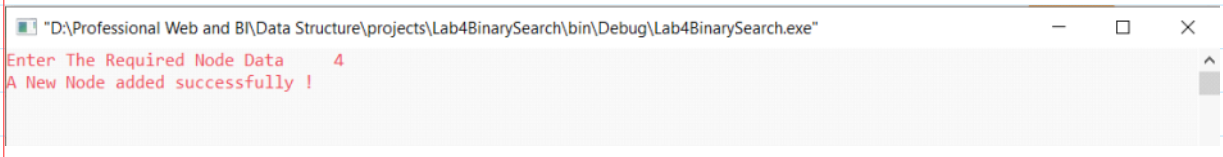
10 December 2022 10:11

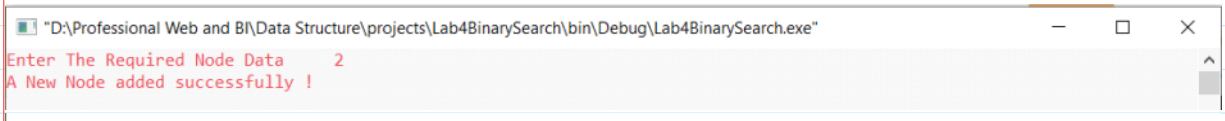
Q1 : Implement a binary Search Tree

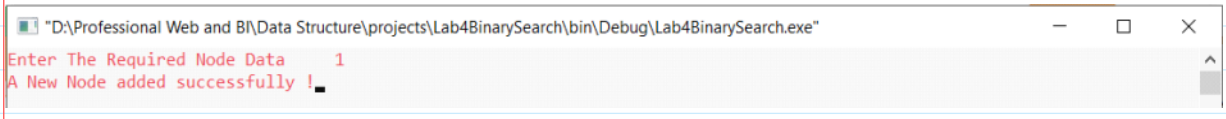
Data Structures & Algorithms
Lab 4
Karim Ali Ramadan
Lab 120

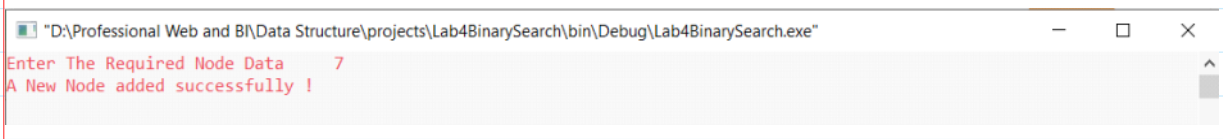


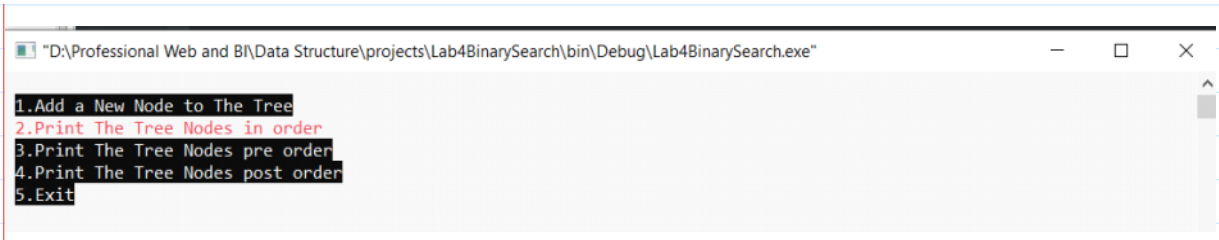




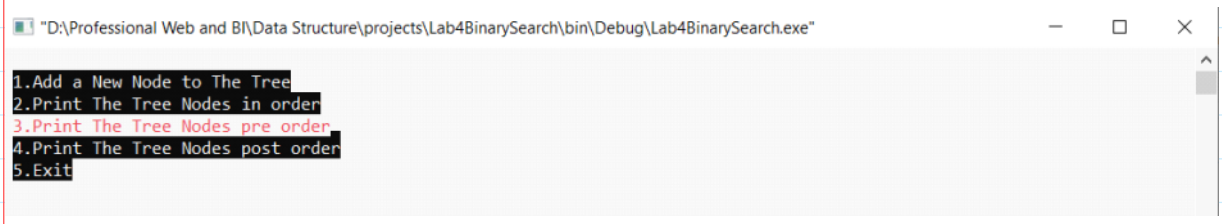


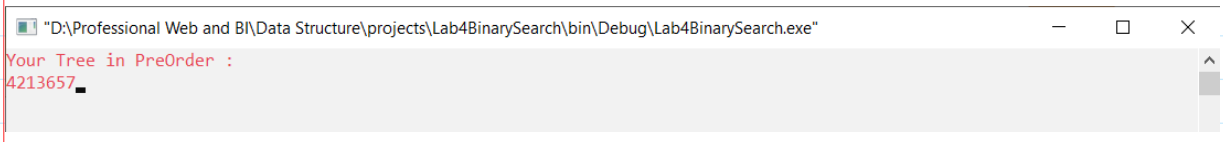


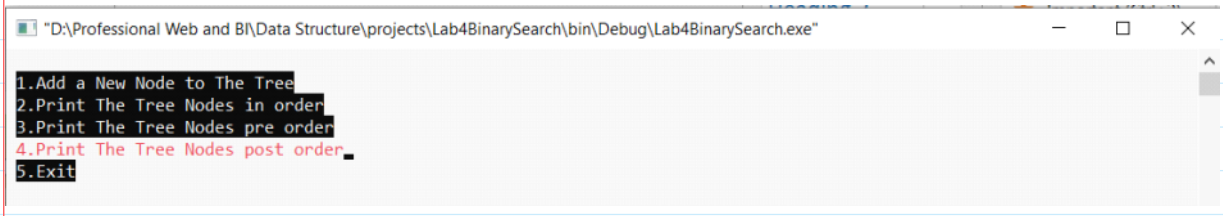



"D:\Professional Web and BI\Data Structure\projects\Lab4BinarySearch\bin\Debug\Lab4BinarySearch.exe"
1.Add a New Node to The Tree
2.Print The Tree Nodes in order
3.Print The Tree Nodes pre order
4.Print The Tree Nodes post order
5.Exit


"D:\Professional Web and BI\Data Structure\projects\Lab4BinarySearch\bin\Debug\Lab4BinarySearch.exe"
Your Tree InOrder :
1234567


"D:\Professional Web and BI\Data Structure\projects\Lab4BinarySearch\bin\Debug\Lab4BinarySearch.exe"
1.Add a New Node to The Tree
2.Print The Tree Nodes in order
3.Print The Tree Nodes pre order
4.Print The Tree Nodes post order
5.Exit


"D:\Professional Web and BI\Data Structure\projects\Lab4BinarySearch\bin\Debug\Lab4BinarySearch.exe"
Your Tree in PreOrder :
4213657


"D:\Professional Web and BI\Data Structure\projects\Lab4BinarySearch\bin\Debug\Lab4BinarySearch.exe"
1.Add a New Node to The Tree
2.Print The Tree Nodes in order
3.Print The Tree Nodes pre order
4.Print The Tree Nodes post order
5.Exit


"D:\Professional Web and BI\Data Structure\projects\Lab4BinarySearch\bin\Debug\Lab4BinarySearch.exe"
Your Tree in PostOrder :
1325764

1

Saturday, December 10, 2022 11:28 AM

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>


#define yAddNode 1
#define yInOrder 2
#define yPreOrder 3
#define yPostOrder 4
#define yExit 5

#define ArrowUp 72
#define ArrowDown 80
#define Enter 13
#define Escape 27
#define Home 71
#define End 79


//═══════════════ Functions Prototype ═══════════════//
void PrintMenu(int x);
void findColor(int y);

//═══════════════ Binary Search Tree ═══════════════//
struct Node {
    int Data;
    struct Node* pLeft;
    struct Node* pRight;
};
struct Node* CreateNode (int d);
struct Node* InsertNode (struct Node* pNode , int d);
void PreOrder (struct Node* pRoot);
void InOrder (struct Node* pRoot);
void PostOrder (struct Node* pRoot);
```

```c
int main ()
{
// I. Variables Declaration

    int x = 1 , y    , App = 1 , d ;
    char Ascii;
    struct Node* pRoot = NULL;



// II.Expressions & Function Calling
    clrscr();

    PrintMenu(x);
    y = 1;
    gotoxy(x,y);
    findColor(y);
    while (App ≠ 0)
    {

        Ascii = getch(); // Normal Key
        if (Ascii == 0) Ascii = getch(); // Extended Key
        switch (Ascii)
        {
            case ArrowUp:
                if (y == yAddNode)
                {
                    y = yExit;
                    PrintMenu(x);
                    gotoxy(x,y);
                    findColor(y);
                }
                else
                {
                    y--;
                    PrintMenu(x);
                    gotoxy(x,y);
                    findColor(y);
                }
                break;
            case ArrowDown:
                if (y == yExit)
                {
                    y = yAddNode;
```

```
            if (y == yExit)
            {
                y = yAddNode;
                PrintMenu(x);
                gotoxy(x,y);
                findColor(y);
            }
            else
            {
                y++;
                PrintMenu(x);
                gotoxy(x,y);
                findColor(y);
            }
            break;
        case Enter:
            switch (y)
            {
                case yAddNode:
                    clrscr();
                    printf("Enter The Required Node Data     ");
                    scanf("%d",&d);
                    pRoot = InsertNode(pRoot , d);
                    printf("A New Node added successfully !");
                    getch();
                    clrscr();
                    break;
                case yInOrder:
                    clrscr();
                    printf("Your Tree InOrder : \n");
                    InOrder(pRoot);
                    getch();
                    clrscr();
                    break;
                case yPreOrder :
                    clrscr();
                    printf("Your Tree in PreOrder : \n");
                    PreOrder(pRoot);
                    getch();
                    clrscr();
                    break;
                case yPostOrder:
                    clrscr();
                    printf("Your Tree in PostOrder : \n");
                    PostOrder(pRoot);
                    getch();
                    clrscr();
                    break;
                case yExit:
                    App = 0;
```

```
                        break;
                    case yExit:
                        App = 0;
                        break;
                }
                break;
            case Escape:
                App = 0;
                break;
            case Home:
                y = yAddNode;
                PrintMenu(x);
                gotoxy(x,y);
                findColor(y);
                break;
            case End:
                y = yExit;
                PrintMenu(x);
                gotoxy(x,y);
                findColor(y);
                break;
            }
    }
if (Ascii != Escape && !(Ascii == Enter && y == yExit)) getch();
return 0;
}
```

```
// III.Functions

void PrintMenu(int x)
{
    textcolor(WHITE);
    textbackground(BLACK);
    gotoxy(x,yAddNode);
    cprintf("1.Add a New Node to The Tree");
    gotoxy(x,yInOrder);
    cprintf("2.Print The Tree Nodes in order");
    gotoxy(x,yPreOrder);
    cprintf("3.Print The Tree Nodes pre order");
    gotoxy(x,yPostOrder);
    cprintf("4.Print The Tree Nodes post order");
    gotoxy(x,yExit);
    cprintf("5.Exit");
```

```
    cprintf("4.Print The Tree Nodes post order");
    gotoxy(x,yExit);
    cprintf("5.Exit");
}

void findColor(int y) {
    textcolor(RED);
    textbackground(WHITE);
    switch (y)
                {
                    case yAddNode:
                        cprintf("1.Add a New Node to The Tree");
                        break;
                    case yInOrder:
                        cprintf("2.Print The Tree Nodes in order");
                        break;
                    case yPreOrder:
                        cprintf("3.Print The Tree Nodes pre order");
                        break;
                    case yPostOrder:
                        cprintf("4.Print The Tree Nodes post order");
                        break;
                    case yExit:
                        cprintf("5.Exit");
                        break;
                }
}
struct Node* CreateNode (int d) {
    struct Node* ptr;
    ptr = (struct Node*) malloc( sizeof(struct Node) );
    if (ptr) {
        ptr→Data = d;
        ptr→pLeft = ptr→pRight = NULL;
    }
    return ptr;
}
struct Node* InsertNode (struct Node* pNode , int d) {
    if (pNode == NULL) { // No List
        pNode = CreateNode(d);
    }
    else if (pNode→Data ⩾ d) {
        pNode→pLeft = InsertNode(pNode→pLeft,d);
    } else {
        pNode→pRight = InsertNode(pNode→pRight,d);
    }
    return pNode;
}
void PreOrder (struct Node* pRoot) {
    if (pRoot) {
        printf("%d" , pRoot→Data);
```

```c
void PreOrder (struct Node* pRoot) {
    if (pRoot) {
        printf("%d" , pRoot→Data);
        PreOrder(pRoot→pLeft);
        PreOrder(pRoot→pRight);
    }
}
void InOrder (struct Node* pRoot) {
    if (pRoot) {
        InOrder(pRoot→pLeft);
        printf("%d" , pRoot→Data);
        InOrder(pRoot→pRight);
    }
}
void PostOrder (struct Node* pRoot) {
    if (pRoot) {
        PostOrder(pRoot→pLeft);
        PostOrder(pRoot→pRight);
        printf("%d" , pRoot→Data);
    }
}
```

2

10 December 2022 10:55

Q2 : Draw the tree

pre order	F	A	E	K	C	D	H	G	B
In order	E	A	C	K	F	H	D	B	G