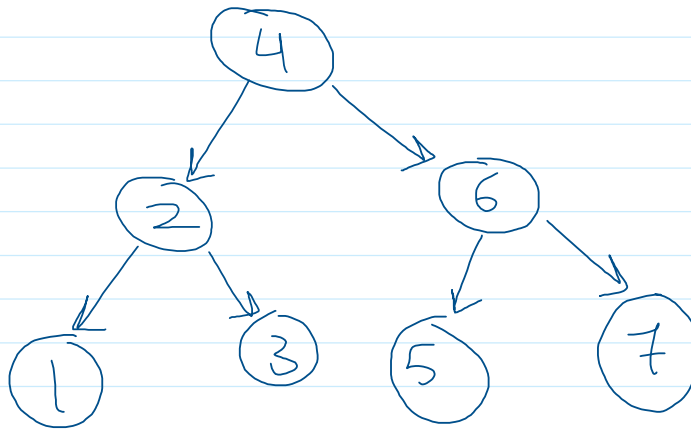


Q1: Implement a binary Search Tree



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"

```
Enter The Required Node Data    4
A New Node added successfully !
```

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

```
Enter The Required Node Data    2
A New Node added successfully !
```

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

```
Enter The Required Node Data    6
A New Node added successfully !
```

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

```
Enter The Required Node Data    1
A New Node added successfully !
```

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

```
Enter The Required Node Data    3
A New Node added successfully !
```

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

```
Enter The Required Node Data    5
A New Node added successfully !
```

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

```
Enter The Required Node Data    7
A New Node added successfully !
```

```
"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  #include <stdio.h>
2  #include <conio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6
7  #define yAddNode 1
8  #define yInOrder 2
9  #define yPreOrder 3
10 #define yPostOrder 4
11 #define yExit 5
12
13 #define ArrowUp 72
14 #define ArrowDown 80
15 #define Enter 13
16 #define Escape 27
17 #define Home 71
18 #define End 79
19
20
21 //===== Functions Prototype =====//
22 void PrintMenu(int x);
23 void findColor(int y);
24
25 //===== Binary Search Tree =====//
26 struct Node {
27     int Data;
28     struct Node* pLeft;
29     struct Node* pRight;
30 };
31 struct Node* CreateNode (int d);
32 struct Node* InsertNode (struct Node* pNode , int d);
33 void PreOrder (struct Node* pRoot);
34 void InOrder (struct Node* pRoot);
35 void PostOrder (struct Node* pRoot);

```



```
1  int main ()
2  {
3  // I. Variables Declaration
4
5      int x = 1 , y , App = 1 , d ;
6      char Ascii;
7      struct Node* pRoot = NULL;
8
9
10
11 // II.Expressions & Function Calling
12     clrscr();
13
14     PrintMenu(x);
15     y = 1;
16     gotoxy(x,y);
17     findColor(y);
18     while (App != 0)
19     {
20
21         Ascii = getch(); // Normal Key
22         if (Ascii == 0) Ascii = getch(); // Extended Key
23         switch (Ascii)
24         {
25             case ArrowUp:
26                 if (y == yAddNode)
27                 {
28                     y = yExit;
29                     PrintMenu(x);
30                     gotoxy(x,y);
31                     findColor(y);
32                 }
33             else
34             {
35                 y--;
36                 PrintMenu(x);
37                 gotoxy(x,y);
38                 findColor(y);
39             }
40             break;
41             case ArrowDown:
42                 if (y == yExit)
43                 {
44                     y = yAddNode;
```

```

42         if (y == yExit)
43         {
44             y = yAddNode;
45             PrintMenu(x);
46             gotoxy(x,y);
47             findColor(y);
48         }
49         else
50         {
51             y++;
52             PrintMenu(x);
53             gotoxy(x,y);
54             findColor(y);
55         }
56         break;
57     case Enter:
58         switch (y)
59         {
60             case yAddNode:
61                 clrscr();
62                 printf("Enter The Required Node Data      ");
63                 scanf("%d",&d);
64                 pRoot = InsertNode(pRoot , d);
65                 printf("A New Node added successfully !");
66                 getch();
67                 clrscr();
68                 break;
69             case yInOrder:
70                 clrscr();
71                 printf("Your Tree InOrder : \n");
72                 InOrder(pRoot);
73                 getch();
74                 clrscr();
75                 break;
76             case yPreOrder :
77                 clrscr();
78                 printf("Your Tree in PreOrder : \n");
79                 PreOrder(pRoot);
80                 getch();
81                 clrscr();
82                 break;
83             case yPostOrder:
84                 clrscr();
85                 printf("Your Tree in PostOrder : \n");
86                 PostOrder(pRoot);
87                 getch();
88                 clrscr();
89                 break;
90             case yExit:
91                 App = 0;

```



```

89             break;
90         case yExit:
91             App = 0;
92             break;
93     }
94     break;
95 case Escape:
96     App = 0;
97     break;
98 case Home:
99     y = yAddNode;
100    PrintMenu(x);
101    gotoxy(x,y);
102    findColor(y);
103    break;
104 case End:
105     y = yExit;
106     PrintMenu(x);
107     gotoxy(x,y);
108     findColor(y);
109     break;
110 }
111 }
112 if (Ascii != Escape && !(Ascii == Enter && y == yExit)) getch();
113 return 0;
114 }

```

```

1 // III.Functions
2
3 void PrintMenu(int x)
4 {
5     textcolor(WHITE);
6     textbackground(BLACK);
7     gotoxy(x,yAddNode);
8     cprintf("1.Add a New Node to The Tree");
9     gotoxy(x,yInOrder);
10    cprintf("2.Print The Tree Nodes in order");
11    gotoxy(x,yPreOrder);
12    cprintf("3.Print The Tree Nodes pre order");
13    gotoxy(x,yPostOrder);
14    cprintf("4.Print The Tree Nodes post order");
15    gotoxy(x,yExit);
16    cprintf("5.Exit");

```

```

14     cprintf("4.Print The Tree Nodes post order ");
15     gotoxy(x,yExit);
16     cprintf("5.Exit");
17 }
18
19 void findColor(int y) {
20     textcolor(RED);
21     textbackground(WHITE);
22     switch (y)
23     {
24         case yAddNode:
25             cprintf("1.Add a New Node to The Tree");
26             break;
27         case yInOrder:
28             cprintf("2.Print The Tree Nodes in order");
29             break;
30         case yPreOrder:
31             cprintf("3.Print The Tree Nodes pre order");
32             break;
33         case yPostOrder:
34             cprintf("4.Print The Tree Nodes post order");
35             break;
36         case yExit:
37             cprintf("5.Exit");
38             break;
39     }
40 }
41 struct Node* CreateNode (int d) {
42     struct Node* ptr;
43     ptr = (struct Node*) malloc( sizeof(struct Node) );
44     if (ptr) {
45         ptr->Data = d;
46         ptr->pLeft = ptr->pRight = NULL;
47     }
48     return ptr;
49 }
50 struct Node* InsertNode (struct Node* pNode , int d) {
51     if (pNode == NULL) { // No List
52         pNode = CreateNode(d);
53     }
54     else if (pNode->Data >= d) {
55         pNode->pLeft = InsertNode(pNode->pLeft,d);
56     } else {
57         pNode->pRight = InsertNode(pNode->pRight,d);
58     }
59     return pNode;
60 }
61 void PreOrder (struct Node* pRoot) {
62     if (pRoot) {
63         printf("%d" , pRoot->Data);

```



```
61 void PreOrder (struct Node* pRoot) {
62     if (pRoot) {
63         printf("%d" , pRoot→Data);
64         PreOrder(pRoot→pLeft);
65         PreOrder(pRoot→pRight);
66     }
67 }
68 void InOrder (struct Node* pRoot) {
69     if (pRoot) {
70         InOrder(pRoot→pLeft);
71         printf("%d" , pRoot→Data);
72         InOrder(pRoot→pRight);
73     }
74 }
75 void PostOrder (struct Node* pRoot) {
76     if (pRoot) {
77         PostOrder(pRoot→pLeft);
78         PostOrder(pRoot→pRight);
79         printf("%d" , pRoot→Data);
80     }
81 }
```

Q2 : Draw the tree

		^R	^R	^L			^R	^L		^R	
Pre order	F	A	E	K	C	D	H	G	B		
In order	^L E	^R A	^{R_i} C	K	^R F	^L H	^R D	^{R_i} B	^R G		

$\underbrace{\hspace{10em}}_L$
 $\underbrace{\hspace{10em}}_{R_i}$

