UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECJ2013 DATA STRUCTURE AND ALGORITHM
SECTION 6
LAB EXERCISE 4

Name	Brendan Dylan Gampa Anak Joseph Dusit @ Dusit (A20EC0021)
Matric No	A20EC0021

LECTURER: TS. DR. JOHANNA BINTI AHMAD

SUBMISSION DATE: 22th JANUARY 2022

1. Given Program 1 and array[ ] in Figure B1, answer the following questions.

```
1   //Program 1
2   using namespace std;
3   int main() {
4   cout<<"Enter The Size Of Array:    ";
5   int size;
6   cin>>size;
7   int array[size], key,i;
8   // Taking Input In Array
9    for(int j=0;j<size;j++){
10   cout<<"Enter "<<j<<" Element: ";
11   cin>>array[j];
12   }
13  //Your Entered Array Is
14   for(int a=0;a<size;a++){
15      cout<<"array[ "<<a<<" ]  =  ";
16      cout<<array[a]<<endl;
17   }
18   cout<<"Enter Key To Search  in Array";
19   cin>>key;
20      for(i=0;i<size;i++){
21        if(key==array[i]){
22    cout<<"Key Found At Index Number :  "<<i<<endl;
23    break;
24      }
25   }
26
27
28  if(i != size){
29  cout<<"KEY FOUND at index :  "<<i;
30  }
31  else{
32  cout<<"KEY NOT FOUND in Array  ";
33  }
33     return 0;
34  }
```

3	2	4	1	5

Figure B1: array [ ]

(a) Name the **searching technique** in Program 1.

Binary searching technique

(b) What is the **complexity time** and **number of comparisons** when searching on array[ ] if the search_key is 10?

Complexity time = O(6)
Number of comparisons = 5

2. Given a search function in Program 2. Answer all the following questions based on INPUT array shown in Figure B2.

```
//Program 2
int search( int  search key, int array size, const int INPUT[] )

{   bool found = false;
    int index = -1 //-1 means record not found
      int MIDDLE, LEFT = 0,
      RIGHT = arraysize-1;

 while ((LEFT <= RIGHT ) && (!found))
 {    MIDDLE = (LEFT + RIGHT )/ 2;    // Get middle index
      if (INPUT[MIDDLE] == search key)
      {      index = MIDDLE;
             found = true;
      }
      else if (INPUT[MIDDLE] > search key)
             RIGHT = MIDDLE - 1;    // search is focused on the left
                                    // side of list
      else
             LEFT = MIDDLE + 1;      // search is focused on the right
                                   / / side of the list
 } //end while

 return index;
}//end function
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
5	9	19	25	34	40	45	49	66	75	88	100

Figure B2: INPUT array

Trace the value of **LEFT, RIGHT, MIDDLE, INPUT**[MIDDLE] and found (as in Table B1) for binary search operation performed onto **INPUT array** with the key numbers being search as following:

1. Search Key=40

Table B1

LEFT	RIGHT	MIDDLE	INPUT[MIDDLE]	found
0	11	5	40	TRUE

2. Search Key=100

Table B1

LEFT	RIGHT	MIDDLE	INPUT[MIDDLE]	found
0	11	5	40	FALSE
6	11	10	88	FALSE
11	11	11	100	TRUE

3. Search Key=8

Table B1

LEFT	RIGHT	MIDDLE	INPUT[MIDDLE]	found
0	11	5	40	FALSE
0	4	2	19	FALSE
0	1	0	5	FALSE
1	1	1	9	FALSE
1	0	-	-	FALSE

3. Tree

d) Give the inorder, preorder and postorder traversal of the tree in Figure 7. [6 marks]

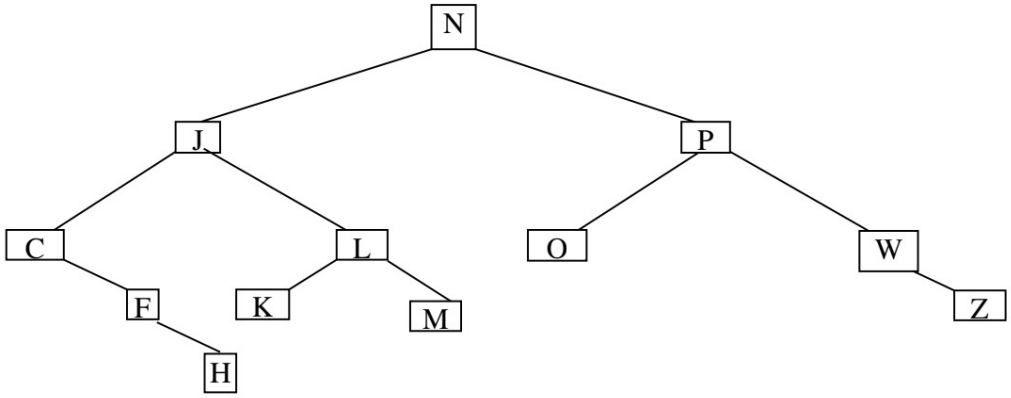


Figure 7: Binary Search Tree of char value

INORDER = H, F, C, J, K, L, M, N, O, P, W, Z

PREORDER = N, J, C, F, H, L, K, M, P, O, W, Z

POSTORDER = H, F, C, K, M, L, J, O, Z, W, P, N

4. Tree

Given the following binary search tree in Figure 8, answer question 5 and 6.

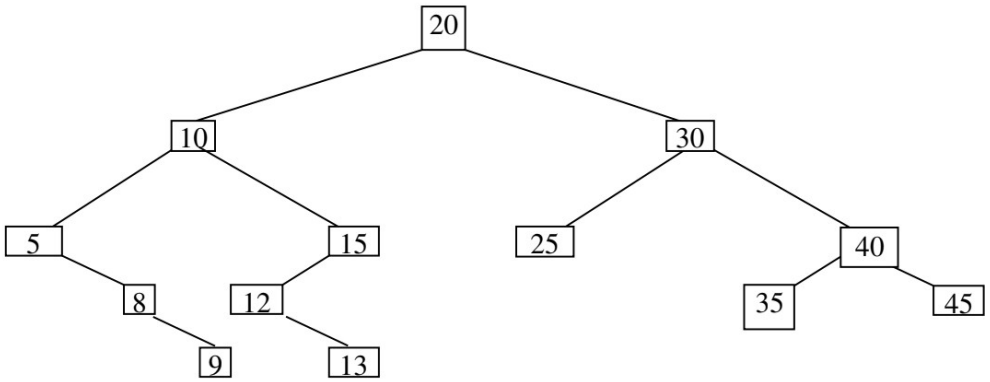


Figure 8: Binary Search Tree of int value

e) Redraw the tree in Figure 8 after nodes with values **18, 16, 19, 33 and 43** are inserted in sequence. [2 marks]

f) Redraw the tree in Figure 8 after the following nodes are deleted **in sequence**. Show the new tree after every deletion.

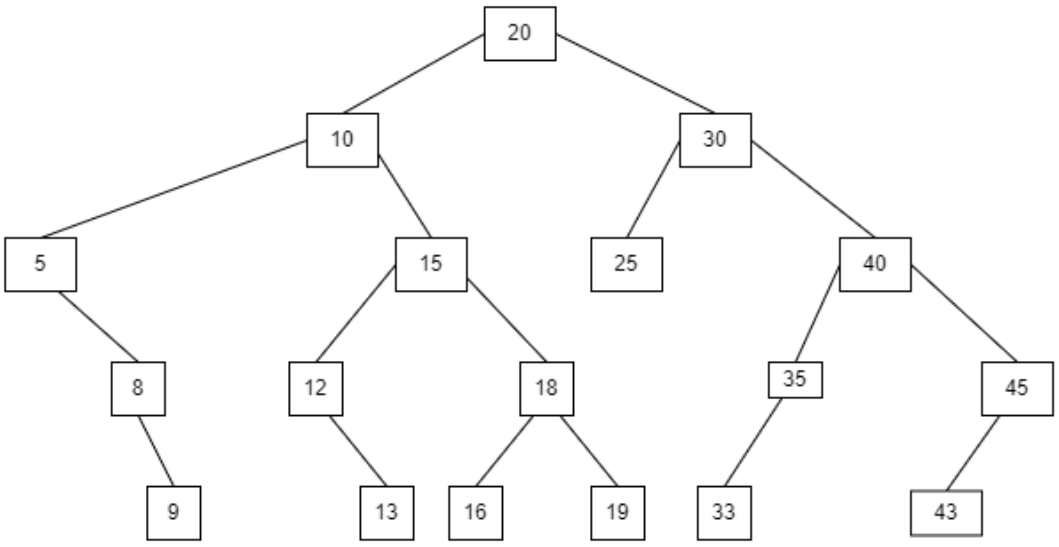
i) 25

ii) 20

iii) 10

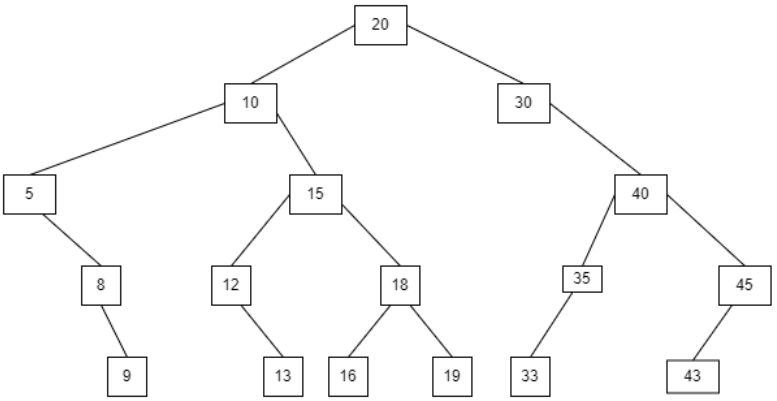
[4 marks]

QUESTION e

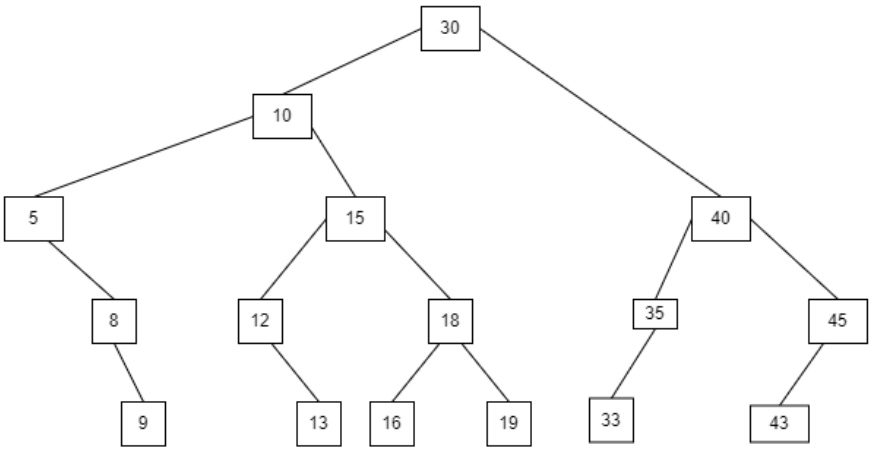


QUESTION f

i) 25



ii) 20



iii) 10