Dr. D.Y.Patil Arts, Commerce &amp;amp; Science College,

Pimpri, Pune-18

S.Y.B.Sc(Comp. Sci) sem-IV 2022-23

Data Structures and Algorithms – II

Practical Assignment 2: Binary Tree Applications

Date: 16/3/23

Set A

a) Write a C program which uses Binary search tree library and displays nodes at each level, count of node at each level and total levels in the tree.

```
#include<stdio.h>
#include<stdlib.h>

int height;

struct node
{
    struct node *lchild;
    int data;
    struct node *rchild;
};
typedef struct node NODE;

NODE *getnode()
{
    NODE *temp;
    temp=(NODE*)malloc(sizeof(NODE));
    printf("\n\n Enter the data : ");
    scanf("%d",&temp->data);
    temp->lchild=NULL;
    temp->rchild=NULL;
    return(temp);
}

NODE *create()
{
    NODE *temp,*ptr,*root;
    char ch;
    root=NULL;
    do
    {
         temp=getnode();
         if(root==NULL)
              root=temp;
         else
         {
              ptr=root;
              while(ptr!=NULL)
```

```
            {
                if(temp->data<ptr->data)
                {
                    if(ptr->lchild==NULL)
                    {
                        ptr->lchild=temp;
                        break;
                    }
                    else
                        ptr=ptr->lchild;
                }
                else
                {
                    if(ptr->rchild==NULL)
                    {
                        ptr->rchild=temp;
                        break;
                    }
                    else
                        ptr=ptr->rchild;
                }
            }//while
        } //else
        printf("\n Add More (Y/N)? : ");
        scanf(" %c",&ch);
    }while(ch=='Y' || ch=='y');
    return(root);
}
int tree_height(NODE * ptr)
{
    if (!ptr)
        return 0;
    else {
        int left_height = tree_height(ptr->lchild);
        int right_height = tree_height(ptr->rchild);
        if (left_height >= right_height)
            return left_height + 1;
        else
            return right_height + 1;
    }
}

void print_level(NODE * ptr, int level)
{
    if (!ptr)
        return;
    if (level == 0)
    {

        printf("%d -> ", ptr->data);
    }
    else
```

```
     {

          print_level(ptr->lchild, level - 1);
          print_level(ptr->rchild, level - 1);
     }
}

void print_tree_level_order(NODE* ptr)
{
     int i;
     if (!ptr)
          return;
     for (i=0; i<height; i++)
     {
          printf("\nLevel %d: ", i);
          print_level(ptr, i);
          printf("\n");
     }
     printf("\n\n-----Complete Level Order Traversal:-----\n");
     for (i=0; i<height; i++)
     {
          print_level(ptr, i);
     }
     printf("\n");
}
int countnodelevel(NODE *ptr,int level)
{
     if(ptr==NULL)
          return 0;
     if(level==0)
          return 1;
     return countnodelevel(ptr->lchild,level-1) +
countnodelevel(ptr->rchild,level-1);

}
main()
{
     NODE *root;
     int i;
     root=create();
     printf("\n");
        height=tree_height(root);
     printf("\nTotal Levels in the tree: %d",height);
     print_tree_level_order(root);
     for (i=0; i<height; i++)
          printf("\nNumber of nodes at [ %d ] Level ::
%d\n",i,countnodelevel(root,i));
}
/*
[root@localhost ass2]# cc levelnodes.c
[root@localhost ass2]# ./a.out
```

```
 Enter the data : 10

 Add More (Y/N)? : y


 Enter the data : 20

 Add More (Y/N)? : y


 Enter the data : 6

 Add More (Y/N)? : y


 Enter the data : 30

 Add More (Y/N)? : y


 Enter the data : 2

 Add More (Y/N)? : y


 Enter the data : 9

 Add More (Y/N)? : n


Total Levels in the tree: 3
Level 0: 10 ->

Level 1: 6 -> 20 ->

Level 2: 2 -> 9 -> 30 ->


-----Complete Level Order Traversal:-----
10 -> 6 -> 20 -> 2 -> 9 -> 30 ->

Number of nodes at [ 0 ] Level :: 1

Number of nodes at [ 1 ] Level :: 2

Number of nodes at [ 2 ] Level :: 3

*/
```

Set B

a) Write a program to sort n randomly generated elements using Heapsort method.

```
// Heap Sort in C
#include <stdio.h>
void swap(int* a, int* b)
{
        int temp = *a;
        *a = *b;
        *b = temp;
}
void heapify(int arr[], int N, int i)
{
        int largest = i;
        int left = 2 * i + 1;
        int right = 2 * i + 2;
        if (left < N && arr[left] > arr[largest])
                largest = left;
        if (right < N && arr[right] > arr[largest])
                largest = right;
        if (largest != i)
         {
                swap(&arr[i], &arr[largest]);
                heapify(arr, N, largest);
        }
}
void printheap(int arr[], int N)
{
int i;
      for (i = 0; i < N; i++)
            printf("%d ", arr[i]);
      printf("\n");
}


void heapsort(int arr[], int N)
{
int i,pass=1;
        // Build max heap
        for (i = N / 2 - 1; i >= 0; i--)
                heapify(arr, N, i);
      printf("\nArray After Building Max Heap:  ");
      printheap(arr, N);
```

```
        // swap 1st and last element
        for (i = N - 1; i >= 0; i--)
         {
                 swap(&arr[0], &arr[i]);
             if(pass<N)
            {
                  printf("\nSorted array after Pass  %d:  ",pass++);
                  printheap(arr, N);
            }
            heapify(arr, i, 0);
        }
}



int main()
{
        int arr[] = {26,5,77,1,61,11,59,15};
        int N = sizeof(arr) / sizeof(arr[0]);
        heapsort(arr, N);
        printf("Sorted array is\n");
        printheap(arr, N);
}

/*
[root@localhost ass2]# cc heapsort.c
[root@localhost ass2]# ./a.out

Array After Building Max Heap:  77 61 59 15 5 11 26 1

Sorted array after Pass  1:  1 61 59 15 5 11 26 77

Sorted array after Pass  2:  26 15 59 1 5 11 61 77

Sorted array after Pass  3:  11 15 26 1 5 59 61 77

Sorted array after Pass  4:  5 15 11 1 26 59 61 77

Sorted array after Pass  5:  1 5 11 15 26 59 61 77

Sorted array after Pass  6:  1 5 11 15 26 59 61 77

Sorted array after Pass  7:  1 5 11 15 26 59 61 77
Sorted array is
1 5 11 15 26 59 61 77
*/
```

Set C

a) Which data structure will be required to display nodes of BST depth wise?
b) Write a C program to displays nodes of BST depth wise.
c) Write a C program to compare two binary search trees (node data wise comparison).
d) How to implement mirror() and copy() functions without recursion?
e) How to convert singly linked list to binary search tree?