USN | 1 | S | I | | | | | | | |

3CCI02

Siddaganga Institute of Technology, Tumkur – 572 103

Department of Computer Science and Engineering

Data structures (3CCI02)

Tutorial-10

Semester : III Section: ' '

Academic year:

Answer the following Questions:

1. Write a C program to implement DEQ using SLL.INclude follwing routines
   a) INsert_front() & Insert_rear()
   b) Delete_front() & Delete_rear()
2. Write a C program to implement Ascending Priority Queue(APQ) using Ordered LInked List.
3. Write a C program to implement Descending Priority Queue(DPQ) using Ordered Linked List.
4. Write a C program to implement APQ & DPQ using SLL with Header node.

Tutorial-11

1. Write a C program to implement APQ & DPQ using DLL with Header node.
2. **What does the following function do for a given Linked List?**

```
void fun1(struct node* head)
{
        if(head == NULL)
                return;

        fun1(head->next);
        printf("%d ", head->data);
}
```

3. **What does the following function do for a given Linked List ?**

```
void fun2(struct node* head)
{
if(head== NULL)
return;
printf("%d ", head->data);

if(head->next != NULL )
fun2(head->next->next);
printf("%d ", head->data);
}
```

4. Given two Strings, represented as linked lists (every character is a node in linked list). Write a function compare() that works similar to strcmp(), i.e., it returns 0 if both strings are same, 1 if first linked list is lexicographically greater, and -1 if second string is lexicographically greater.

USN | 1 | S | I | | | | | | | |

3CCI02

Siddaganga Institute of Technology, Tumkur – 572 103

Department of Computer Science and Engineering

Data structures (3CCI02)

Tutorial-12

Semester : III Section: ' '

Academic year:

Answer the following Questions:

1. Unlike C++ and Java, C doesn't support generics. How to create a linked list in C that can be used for any data type? In C, we can use void pointer and function pointer to implement the same functionality. The great thing about void pointer is it can be used to point to any data type. Also, size of all types of pointers is always is same, so we can always allocate a linked list node. Function pointer is needed to process actual content stored at address pointed by void pointer.
2. Given a linked list and two keys in it, swap nodes for two given keys. Nodes should be swapped by changing links. Swapping data of nodes may be expensive in many situations when data contains many fields.

   It may be assumed that all keys in linked list are distinct.

Tutorial-13

1. Check if a given array can represent Preorder Traversal of Binary Search Tree.Given an array of numbers, return true if given array can represent preorder traversal of a Binary Search Tree, else return false.

Examples:

Input: pre[] = {2, 4, 3}
Output: true
Given array can represent preorder traversal
of below tree

```
  2
   \
    4
   /
  3
```

Input: pre[] = {2, 4, 1}
Output: false
Given array cannot represent preorder traversal
of a Binary Search Tree.
Input: pre[] = {40, 30, 35, 80, 100}
Output: true
Given array can represent preorder traversal
of below tree

```
     40
   /   \
 30     80
   \     \
   35    100
```

Input: pre[] = {40, 30, 35, 20, 80, 100}
Output: false
Given array cannot represent preorder traversal
of a Binary Search Tree.

USN | 1 | S | I | | | | | | | |

3CCI02

Siddaganga Institute of Technology, Tumkur – 572 103

Department of Computer Science and Engineering

Data structures (3CCI02)

Tutorial-14

Semester : III Section: ' '

Academic year:

Answer the following Question:

Write a program for the follwing solution given

How to efficiently implement k stacks in a single array?

We have discussed space efficient implementation of 2 stacks in a single array. In this , a general solution for k stacks is discussed. Following is the detailed problem statement.

Create a data structure kStacks that represents k stacks. Implementation of kStacks should use only one array, i.e., k stacks should use the same array for storing elements. Following functions must be supported by kStacks.

push(int x, int sn) –> pushes x to stack number 'sn' where sn is from 0 to k-1
pop(int sn) –> pops an element from stack number 'sn' where sn is from 0 to k-1

Method 1 (Divide the array in slots of size n/k)

A simple way to implement k stacks is to divide the array in k slots of size n/k each, and fix the slots for different stacks, i.e., use arr[0] to arr[n/k-1] for first stack, and arr[n/k] to arr[2n/k-1] for stack2 where arr[] is the array to be used to implement two stacks and size of array be n.

The problem with this method is inefficient use of array space. A stack push operation may result in stack overflow even if there is space available in arr[]. For example, say the k is 2 and array size (n) is 6 and we push 3 elements to first and do not push anything to second second stack. When we push 4th element to first, there will be overflow even if we have space for 3 more elements in array.

Method 2 (A space efficient implementation)

The idea is to use two extra arrays for efficient implementation of k stacks in an array. This may not make much sense for integer stacks, but stack items can be large for example stacks of employees, students, etc where every item is of hundreds of bytes. For such large stacks, the extra space used is comparatively very less as we use two *integer* arrays as extra space.

Following are the two extra arrays are used:
1) top[]: This is of size k and stores indexes of top elements in all stacks.
2) next[]: This is of size n and stores indexes of next item for the items in array arr[]. Here arr[] is actual array that stores k stacks.
Together with k stacks, a stack of free slots in arr[] is also maintained. The top of this stack is stored in a variable 'free'.

All entries in top[] are initialized as -1 to indicate that all stacks are empty. All entries next[i] are initialized as i+1 because all slots are free initially and pointing to next slot. Top of free stack, 'free' is initialized as 0.

USN | 1 | S | I | | | | | | | |

3CCI02

Siddaganga Institute of Technology, Tumkur – 572 103

Department of Computer Science and Engineering

Data structures (3CCI02)

Tutorial-15

Semester : III Section: ' '

Academic year:

Answer the following Questions:

1. Find Minimum Depth of a Binary TreeGiven a binary tree, find its minimum depth. The minimum depth is the number of nodes along the shortest path from root node down to the nearest leaf node.

2. Maximum Path Sum in a Binary Tree:Given a binary tree, find the maximum path sum. The path may start and end at any node in the tree.

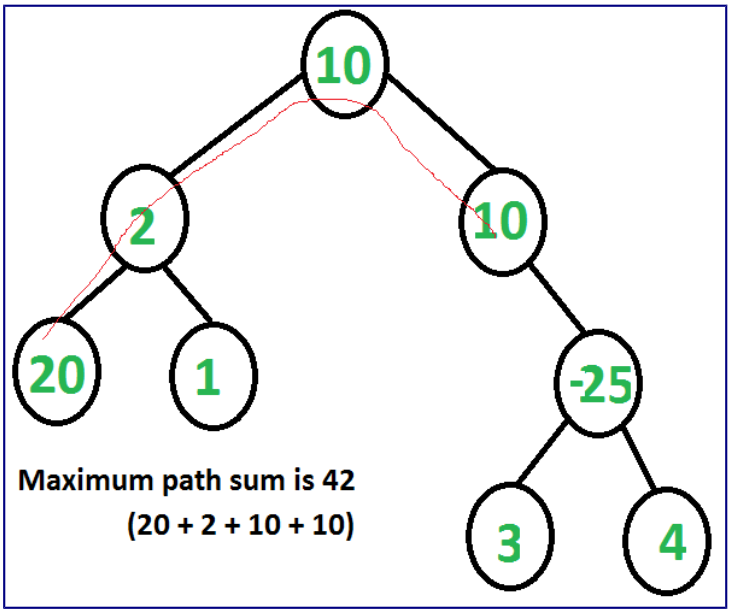
Example:

Input: Root of below tree

```
    1
   / \
  2   3
```

Output: 6

See below diagram for another example.
1+2+3



3. Write a c function to find Inorder predecessor and successor for a given key in BST

4. Write a C program to create Threaded binary tree and do traversal of the tree..