STUDENT PORTFOLIO

Name: Y.BHEEMA SHANKAR CHOWDARY
Register Number: RA2111030010050
Mail ID: yy2460@srmist.edu.in
Department: CSE
Specialization: Cyber Security
Semester: 3rd

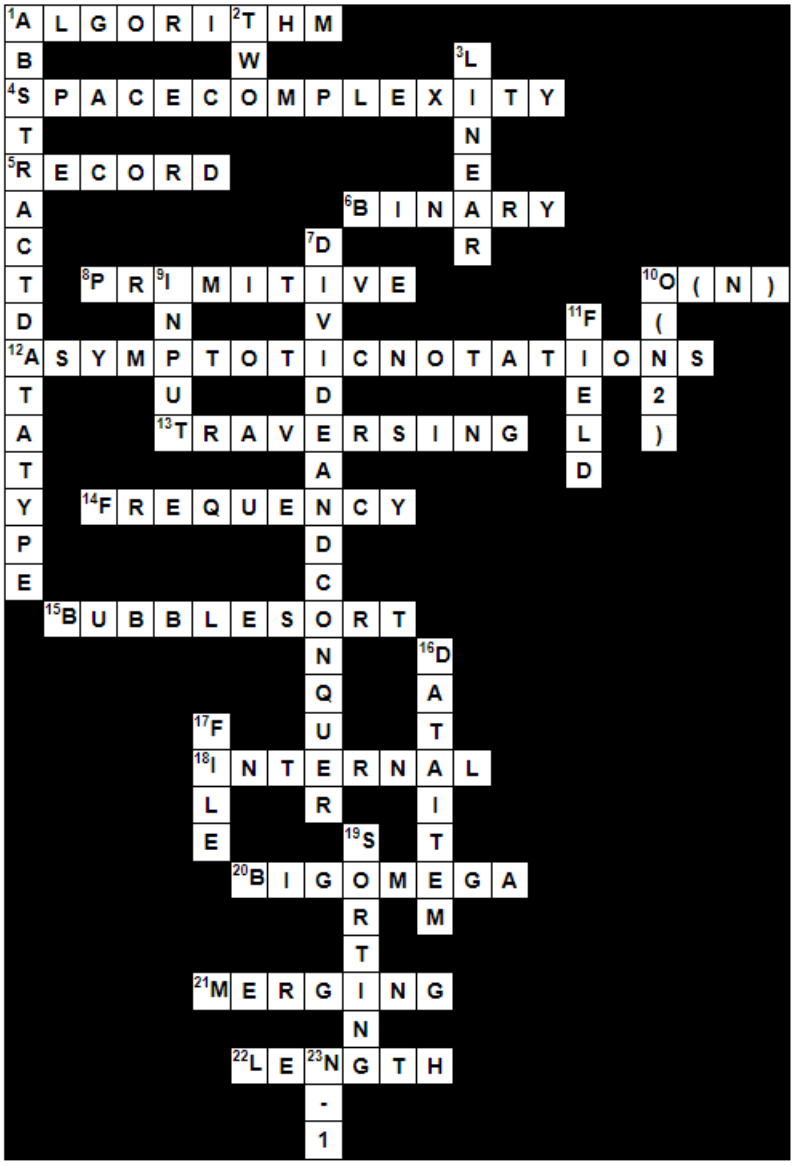
Subject Title: 18CSC201J Data Structures and Algorithms

Handled By: (Dr.M.Jeyaselvi)

Assignment – CrossWord Puzzle (Unit 1,2,3, & 4)

UNIT-1 18CSC201J -DATA STRUCTURES AND ALGORITHMS

Correct! Well done.
Your score is 100%.



Check

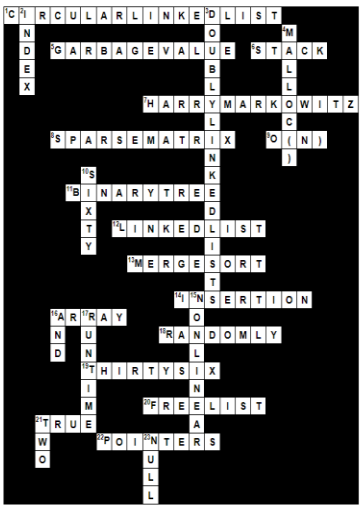
18CSC201J_DATA STRUCTURES AND ALGORITHMS

UNIT II

Correct! Well done.
Your score is 99%.

Across: 12. Polynomial addition can be implemented using _____

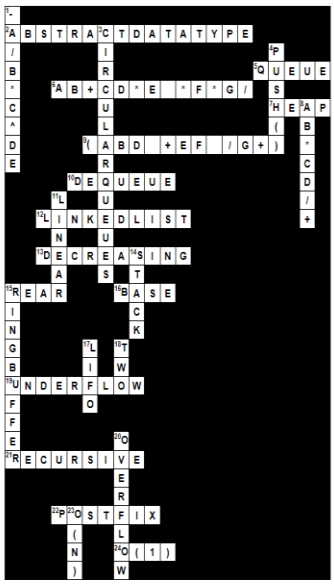
LINKEDLIST Enter



Check

Your score is 98%.
Some of your answers are incorrect. Incorrect squares have been blanked out.

Across: 6. The postfix form of the expression (A+ B)*(C*D- E)*F / G is? AB+CD*E-*F*G/ Enter

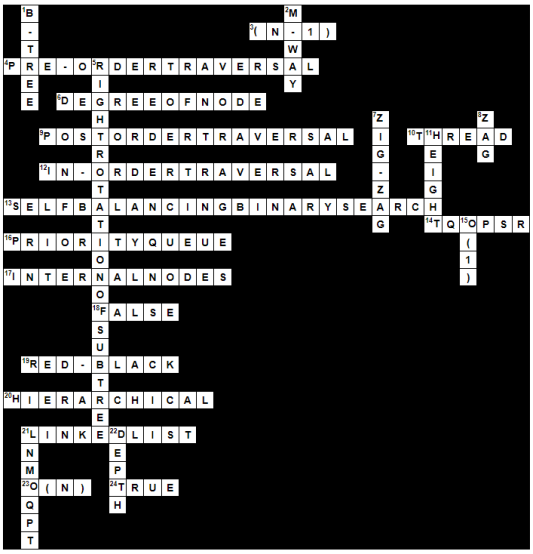


Check

Your score is 97%.
Some of your answers are incorrect. Incorrect squares have been blanked out.

Down: 1: 2-3 tree is a specific form of _____

B-TREE Enter



Check

Assignment

(what is the most interesting part in the assignment)

The most interesting part in the assignment is that's its fun to solve this crossword , This is not boring as other cross word problem

ASSIGNMENT-1

RA2111030010050
Y. Bheema Shankar

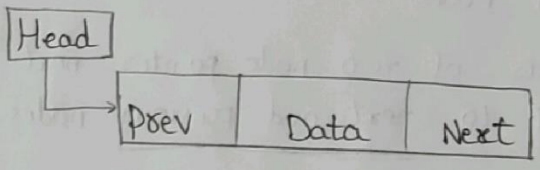
ASSIGNMENT-1

1. Definition of Circular Doubly Linked List?

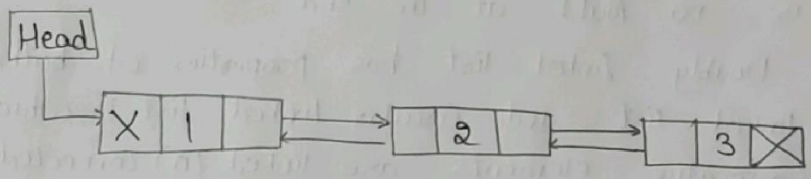
- A doubly linked list is a linked data structure that is represented as a set of nodes.
- A doubly linked list consists of two node pointers next and prev, which point to next and previous nodes respectively
- The circular linked list is a linked list where all nodes are connected to form a circle.
- There is no NULL at the end
- Circular Doubly Linked list has properties of both doubly linked list and Circular linked list in which two consecutive elements are linked (&) connected by the previous and next pointer and the last node points to the first node points to the last node by the previous pointer.
- As doubly circular linked list is a combination of doubly and circular linked list, we can traverse in both directions and also the last node of the list connects to the start node as next, and start node has the last node in the previous pointer.

2. Graphical Representation of Circular Doubly Linked list

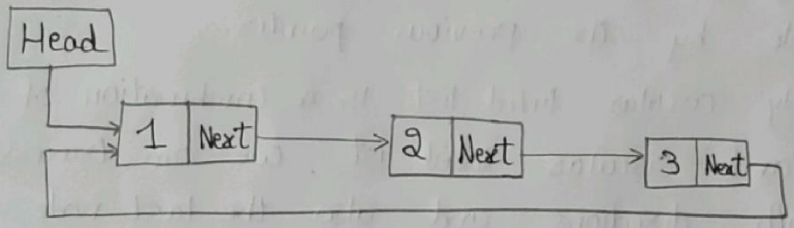
A Doubly Linked List



In doubly linklist a node consists of 3 parts node data, pointer to the nextnode in sequence, Pointer to previous node



Circular Linked List



3. Algorithm for Circular Doubly Linked List

A) Step-1 :- If PTR Null
Write overflow
Go to step 13
[End of if]

Step-2:- SET NEW_NODE = PTR

Step-3 :- SET PTR = PTR → NEXT

Step-4 :- SET NEW_NODE → DATA = VAL

Step-5 :- SET TEMP = HEAD

Step-6 :- Repeat Step 7 while TEMP → NEXT != HEAD

Step-7 :- SET TEMP = TEMP → NEXT
[END of LOOP]

Step-8 :- SET TEMP → NEXT = NEW_NODE

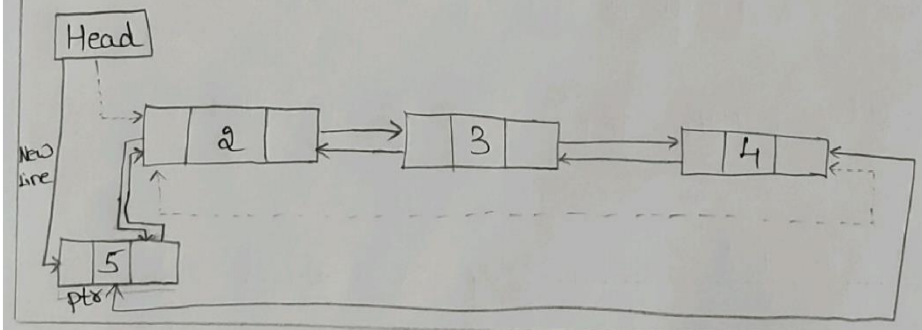
Step-9 :- SET NEW_NODE → PREV = TEMP

Step-10 :- SET NEW_NODE → NEXT = HEAD

Step-11 :- SET HEAD → PREV = NEW_NODE

Step-12 :- SET HEAD = NEW_NODE

Step-13 :- Exit



4. Code for Insertion and Deletion in Circular Doubly linked list.

Ans C program to implement the Circular Doubly linked list.

```
#include <stdio.h>
#include <stdlib.h>
Struct node
{
  Struct node * prev;
  int data;
  Struct node * next;
};
Stouct node * head = NULL;
Sfruct node * create (int);
void insert_begin(int);
void insert_end (int);
void insert_mid (int, int);
Void delete_begin();
void delete_end();
Void delete_mid();
void display ();
int get data();
int get_position();
```

```
int main()
{
  int choice;
  int data, position;
Printf("\n Enter your choice: ");
Scanf ("%d", &choice);
Switch (choice){
 Case 1:
  Printf("\n Inserting a node at beginning");
  data = get_data();
  insert_begin(data);
  break;
 Case 2:
  Printf("\n Inserting a node at end");
   data = get_datal);
   insert_end (data);
   break;
Case 3:-
   printf("\n Inserting a node at the given position");
    data = get_datal);
    Position = get_position();
    insert_mid(position, data);
    break;
```

```
Case 4:
  Printf("\n Deleting a node from beginning\n");
   delete_begin();
   break;
Case 5:
   Printf("\n Deleting a node from end\n");
   delete_end();
   break;

Case 6:
   Printf("\n Delete a node from given position\n");
   Position = get_position();
   delete_mid (Position);
   break;
   default:
      Printf("\n Invalid choice\n");
  }
 Printf("\n Do you want to continue?");
  }
   Scanf
  }
  Stouct node *create (int data)
{
```

```
Stouct node * new_node =(stouct node *) malloc (sizeof
                                        (stoud node));
if (new_node == Null)
 {
  Printf ("In can't be allocated \n");
  return NULL;
 }
new_node --> data = data;
new_node --> next = NuLL;
new_node --> prev = NULL;
   return new_node;
 }
 //inserting node at beginning
 void insert_begin (int data)
 {
   Stouct node * new_node = create (data);
  if (new_node)
  {
   if (head == NuLL)
   {
  new_node --> next = new_node;
  new_node --> prev = new_node;
    head = new_node;
    return ;
  }
```

```
head -> prev -> next = new_node;
new_node -> prev = head -> prev;
new_node -> next = head;
head -> prev = new_node;
head = new_node;
}
}
}
//Inserting at the end
Void insert_end (int data)
{
struct node * new_node = create(data);
if (new_node){
if (head == Null)
{
new_node -> next = new node;
new_node -> prev = new_node;
head = new_node;
return;
}
head -> prev -> next = new_node;
new_node -> prev = head -> prev;
new_node -> next = head;
head -> prev = new_node;
}
}
```

```
//inserting node at given position
void insert_mid(int position, int data)
{ if (position <=0)
  {
    Printf("\n Invalid position \n"); }
else if (head == NULL && position > 1){
    Printf("\n Invalid position\n");
}
else if(head != NULL && position > (large_size)){
  Printf("\n Invalid position\n");
}
else if (position ==1){
      insert_begin(data);
  }
else {
struct node *new_node = create(data);
if(new_node != NULL){
  struct node *temp = head, *prev=NULL;
    int i=1;
while (++i <= position){
```

```
        prev = temp;
        temp = temp -> next;
    }
    prev -> next = new_node;
    new_node -> next = temp;
    }
  }
}

//Deleting a node at beginning
void delete_begin(){
    if(head == NULL){
  Printf("\n List is Empty \n");
    return;
}
else if(head -> next == head){
    free(head);
   head = NULL;
    return;
}
```

```
Stuct node * last_node =head -> prev;
  head -> prev = last_node -> prev;
  free(last_node);   last_node = NULL; }

// Deleting the node from given position
void delete_mid(int position){
  if (position <=0 && position > list.size()){
    printf("\n Invalid position\n");
  }
 else if (position == 1){
      delete_begin();
 else if (position == List_size()) {
      delete_end(); }
 else{
      struct node *temp =head;
      struct node * prev =NULL;
      int i = 1;
while (i< position){
      prev = temp;
      temp = temp -> next;
      it = 1; }
```

```
prev -> next = temp -> next;
temp -> next -> prev = prev;
free (temp);
temp = NULL;
}
//display list
void display() {
if (head == NULL){
Print ("\n List is empty ! \n");
return ;
}
Struct node * temp = head;
do {
Printf ("%d", temp -> data);
temp = temp -> next; }
while (temp! = head);
}
int get_data()
{
int data;
Printf ("\n enter data: \n");
Scanf ("%d", &data);
```

```
return data;}
int get_position(){
    int position;
    Printf("Enter position: ");
    Scanf("%d", &position);
    return position;
}
```

5) Advantages and disadvantages of Circular doubly linked list.

A) Advantages:-

1. If we are at a node, then we can go to any node. But in linear linked list it is not possible to go to previous node

2. It saves time when we have to go to the first node. It can be done in single step because there is no need to traverse in between nodes.

Disadvantages:-

1. It is not easy to reverse the linklist
2. if proper case is not taken, then the problem of infinite loop can occur.
3. If we at a node & go back to the previous node, then we can not do it in single step. Instead we have to complete the entire circle by going through the in between nodes.

Assignment - 2

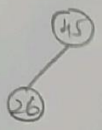
Name:- Y. Bheema Shankar
REG NO:- RA2111030010050

Create a binary search tree for the following numbers start from an empty binary search tree.
45, 26, 10, 60, 70, 30, 40 Delete key 10, 60 and 45 one after the other and show the trees at each stage.

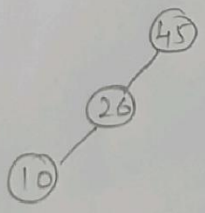
Binary Search Tree

45, 26, 10, 60, 70, 30, 40

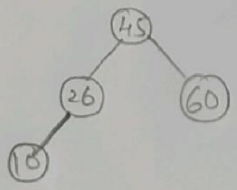
Inserting 26:-

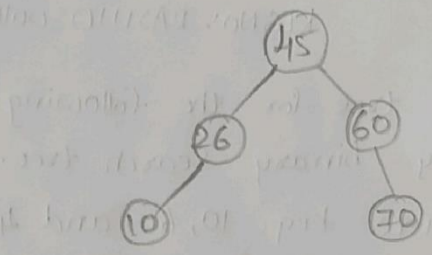


Inserting 10:-

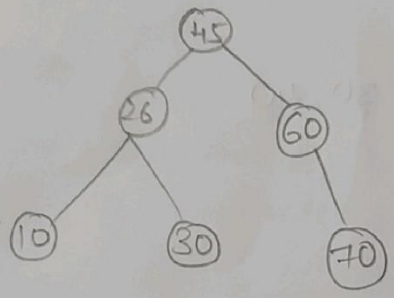


Inserting 60:-

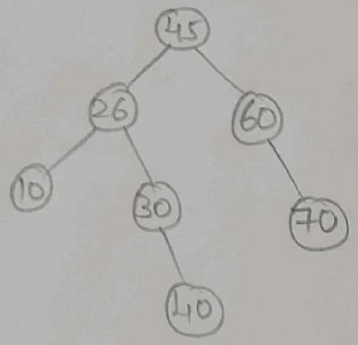




Inserting 30 :-

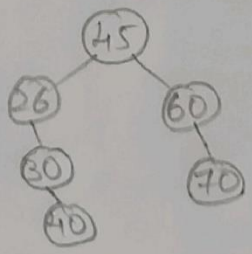


Inserting 40 :-

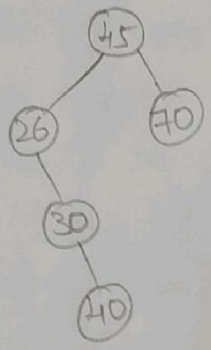


Deleting elements 10, 60, 45

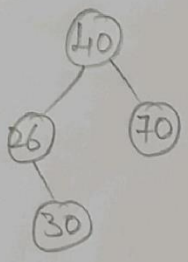
Deleting 10



Deleting 60:-



Deleting 45:-



Binary Search Tree

Definition of Binary Search Tree

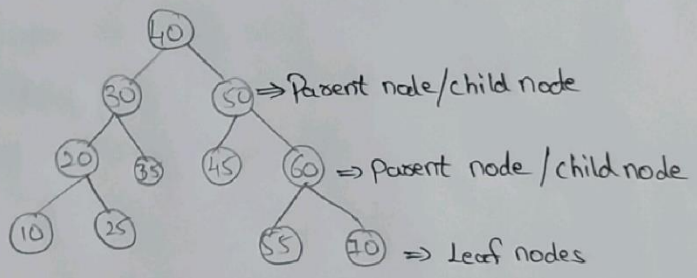
Binary search tree is a node-based binary tree data structure which has following properties

1) The left subtree of a node contains only nodes with keys lesser than the root's key.

2) The right subtree of a node contains only nodes with keys greater than the root's key

3) The left and right subtree each must also be a binary search tree.

Graphical Representation of BST



Algorithm :-

Step - 1 :- Start

Step - 2 :- Create a new node

Step - 3 :- If tree is empty make new node as root

Step - 4 :- Else compare new node with root element and the next nodes if it is greater put it in the right part else put it in the left part.

Step - 5 :- Repeat 2 until user enters 0

Step - 6 :- End

ode :-

```
#include <stdio.h>
#include <stdlib.h>
Struct node {
    int key;
    Struct node * left * right };
    Struct newNode (int item){
    Struct node * temp = (Struct node *)malloc(size of (Struct node));
    temp -> key = item
    temp -> left = NULL;
    temp -> right = NULL;
    return temp; }
    void inorder (struct node * root) {
        if (root != NULL)
        { inorder (root -> left);
          printf ("%d ", root -> key);
          inorder(root -> right);
        }
        }
Struct node * insert (Struct node * node, int key) {
        if (node == NULL)
        return new Node (key);
        if (key < node -> key)
        node -> left = insert (node -> left, key)
```

```
else
    node -> right = insert(node -> right, key);
    return node;
}

struct node * deleteNode(struct node * root, int key) {
    if (root == NULL)
        return root;
    if (key < root -> key)
        root -> left = deleteNode(root -> left, key)
    else if (key > root -> key)
        root -> right = deleteNode(root -> right, key);
    else {
        if (root -> left = NULL) {
            struct node * temp = root -> right;
            free(root);
            return temp;
        }
    else if (root -> right == Null) {
        struct node * temp = root -> left
        free(root);
        return temp;
    }
}
```

```
int main(){
Stouct node * root = NULL;
int n=1, data, x;
while (n! = 0)
Scanf("%d", &data)
{
 root = insert (root, data)
 Printf(" Enter 0 to exit \n");
 Scanf ("%d", &n);
}
 Printf (" Enter node you want to delete");
 Scanf ("%d", &x);
inorder (root);
  root = delete Node (root, x);
inorder (root);
return 0;
}
```

Advantages of BST

1) Binary Search Tree is fast in insertion and deletion when balanced

2) We can also do range - queries. find keys b/w N and M

3) Binary Search Tree is simple as compared to other data structures.

Disadvantages:-

1) The main disadvantage is that we should always implement a balanced binary search tree

2) Accessing the element in BST is slightly slower than array.

3) A BST can be imbalanced or degenerated which can increase the complexity

Assignment - 3

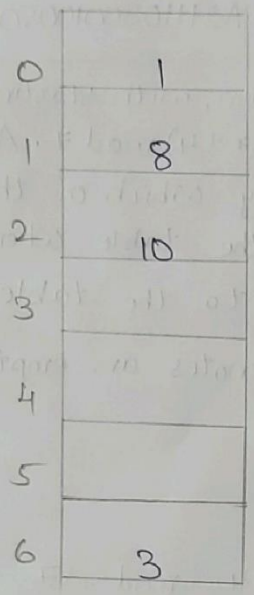
Name :- Y Bheema Shankar Chowdary
RegNO :- RA2111030010050

1. Consider a hash table of size seven, with starting index zero, and a hash function $(3x+4) \bmod 7$. Assuming the hash table is initially empty which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

Sol Given keys :- 1, 3, 8, 10.

$h(x) = 3x+4 \quad \text{mod} = 7$

Key	Location
1	$[3(1)+4] \% 7 = 0$
3	$[3(3)+4] \% 7 = 6$
8	$[3(8)+4] \% 7 = 0$ [Put in the next available space] = 1
10	$[3(10)+4] \% 7 = 6$ [Put in the next available space] = 2

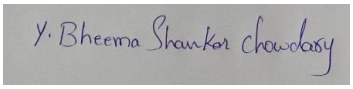


$\Rightarrow$ 1, 8, 10, –, –, —, 3

Therefore answer is option (B)

My code chef:
https://www.codechef.com/users/srmcse_151
MY HACKER RANK:-
https://www.hackerrank.com/yy2460

Any other
(Write if you registered or practise apart from Codechef(ex. Hackerrank, Leetcode etc.)



Signature

Note: Enclose the assignment and relevant certificates along with the profile