STUDENT PORTFOLIO

Name: B Venkata Nikhil Reddy
Register Number:RA2111030010056
Mail ID:bb0377@srmist.edu.in
Department: NWS
Specialization: Cyber Security
Semester:3

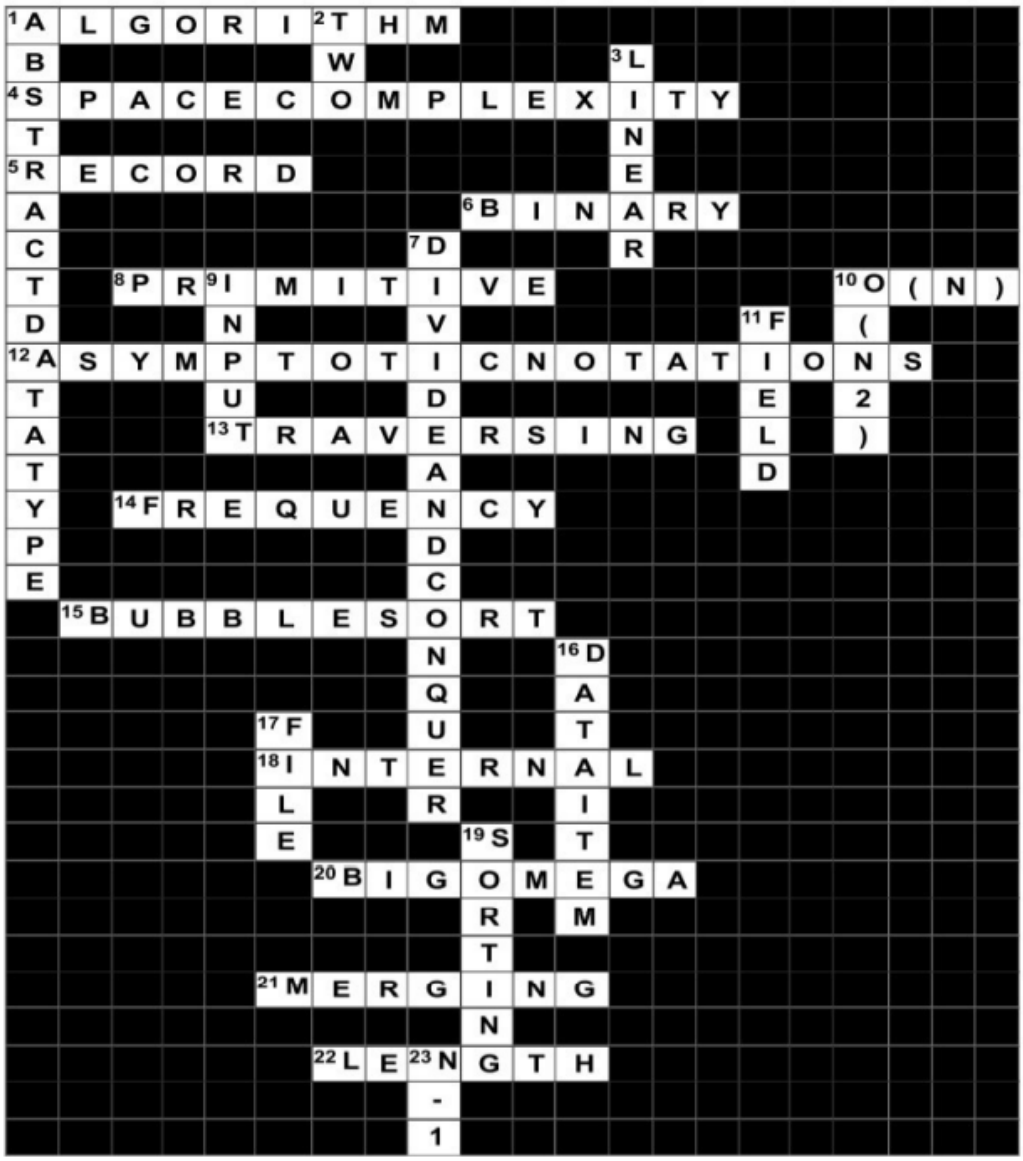
Subject Title: 18CSC201J Data Structures and Algorithms

Handled By: Dr.M.Jeyaselvi

Assignment – CrossWord Puzzle (Unit 1,2,3, & 4)
(Write about the assignment questions and how u solved differently)

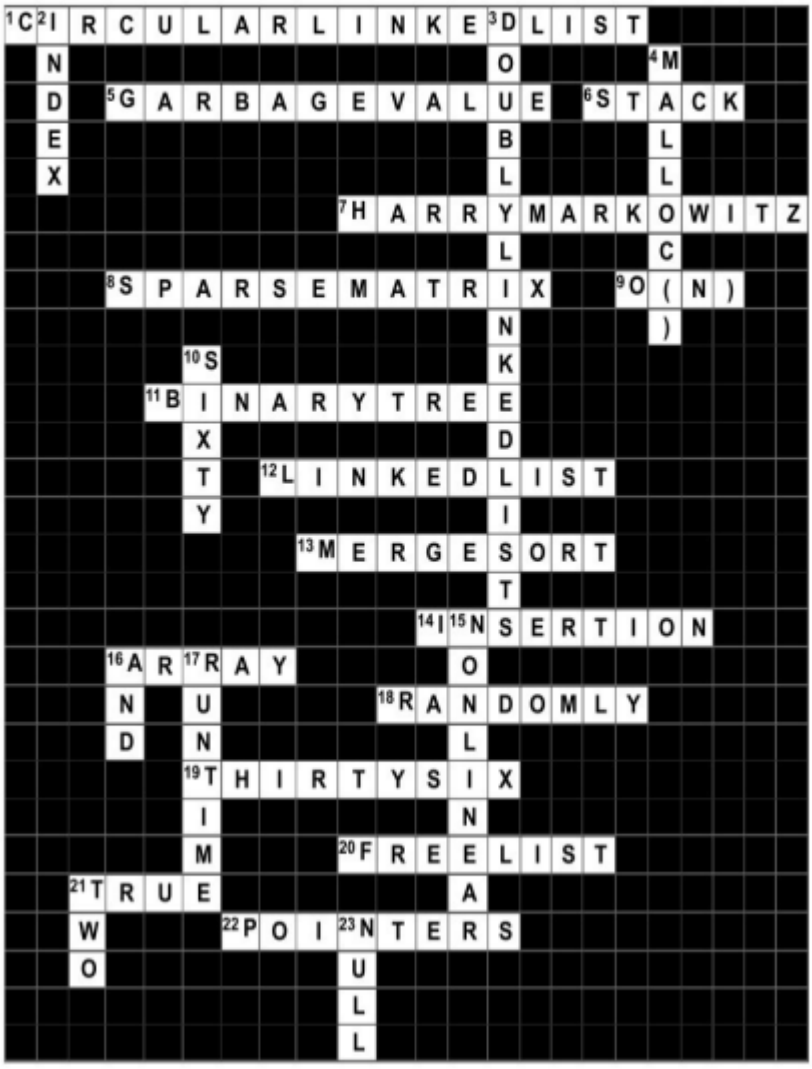
UNIT-1 DATA STRUCTURES

Prepared by Dr. D.SHINY IRENE
AP/CSE/SRMIST



UNIT-2 DATA STRUCTURES & ALGORITHMS

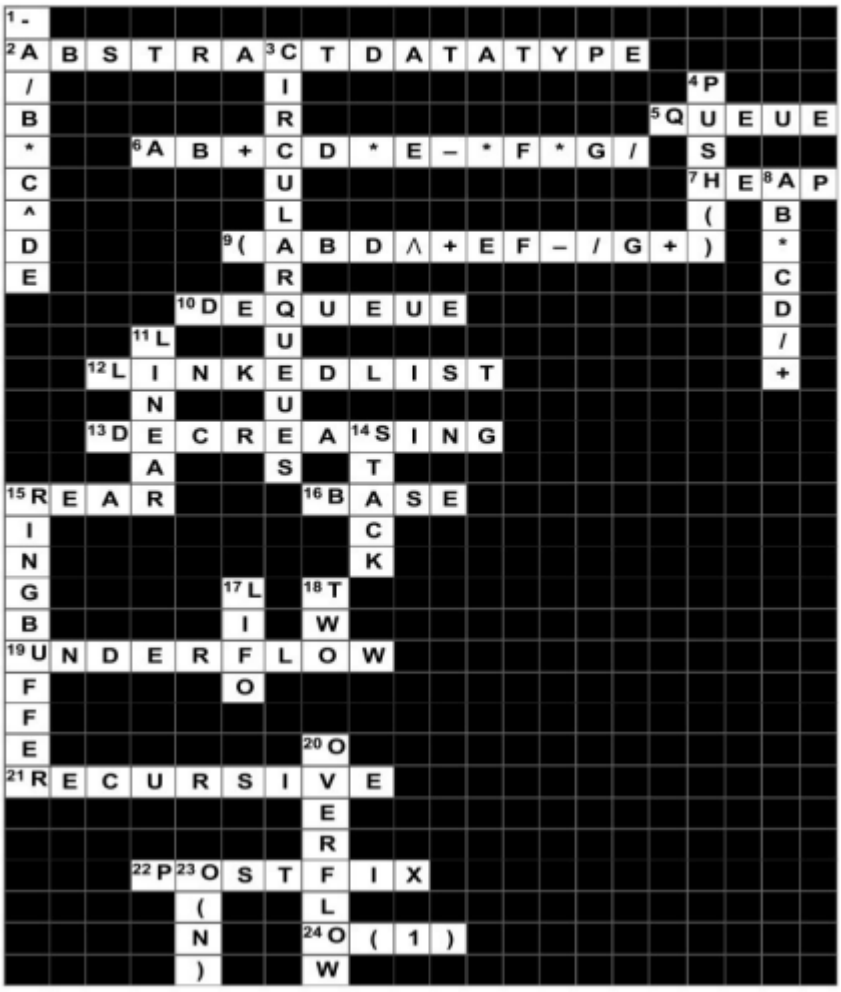
Prepared by Dr. D.SHINY IRENE
AP/CSE/SRMIST



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

UNIT-3 DATA STRUCTURES & ALGORITHMS

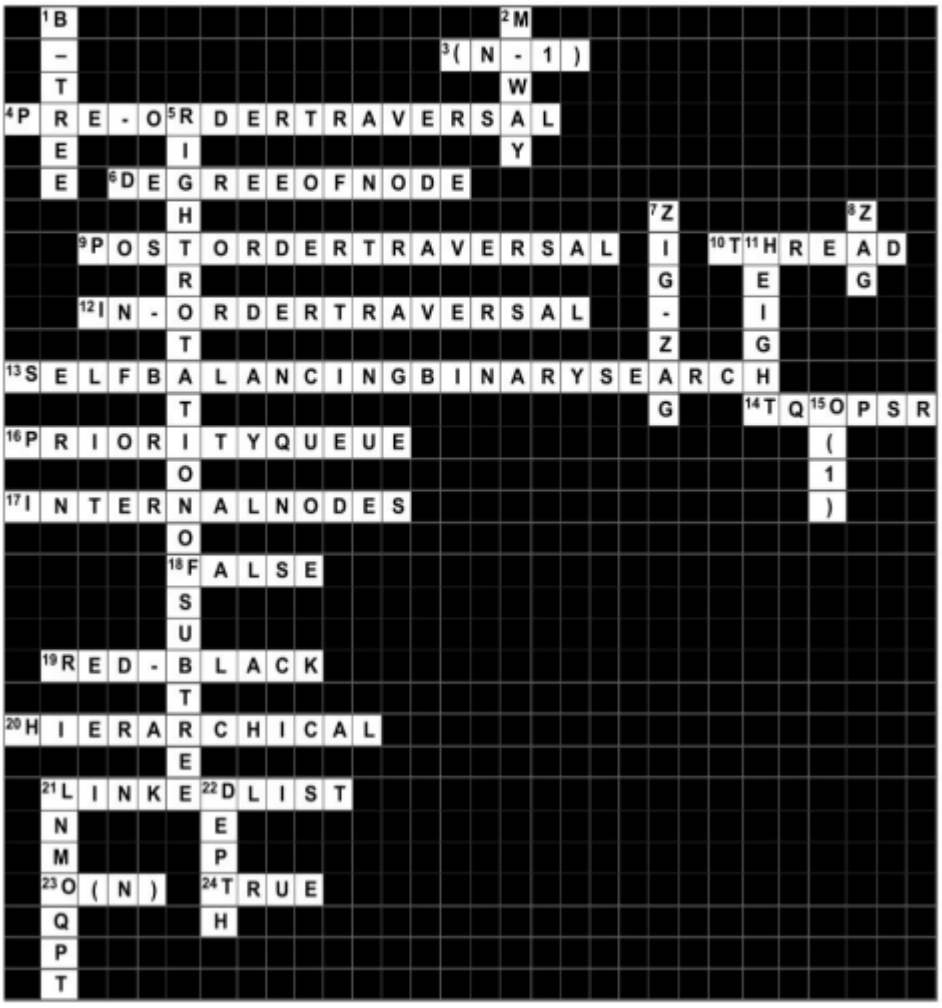
Prepared by Dr. D.SHINY IRENE
AP/CSE/SRMIST



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

UNIT-4 DATA STRUCTURES & ALGORITHMS

Prepared by Dr. D.SHINY IRENE
AP/CSE/SRMIST



Assignment **(what is the most interesting part in the assignment)** **Solving the puzzle was quite good.I was able to recall all the topics and at the same time I was not feeling bored.**

Assignment - I

B.V. Nikhil Reddy
RA2111030010051

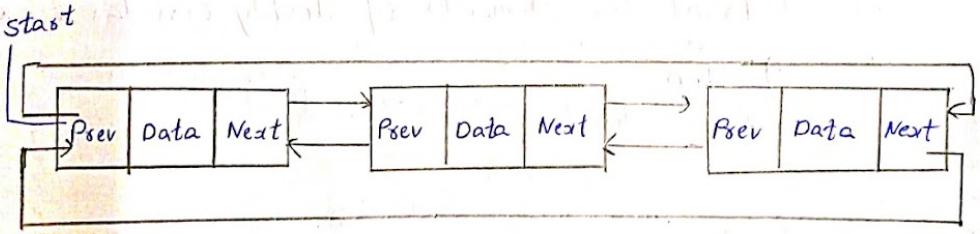
Circular doubly linked list

A circular doubly linked list has the properties of both doubly linked list and circular linked list in which two consecutive elements are linked or connected by the previous and next pointer and last node points to the first node by the next pointer and also the first node points to the last node by the previous pointer

Each nodes consist three fields

1) Previous
2) Data
3) Next

Graphical representation of doubly circular linked list



Algorithm

1) Start

2) Create a structure node with three variables namely

```
struct node *prev;
int data;
struct node *next;
```

3) Create a new node

4) If list is empty assign prev and next to the newnode itself update newnode as head go to step 3

5) Else assign next of new_node as start node and assign prev of new_node as last node

6) Update new node as the last node. goto step 3 until user enters 0

7) Print the elements of doubly circular linked list

8) Stop

Code

```
#include<stdio.h>
#include<stdlib.h>
struct node {
 struct node *prev;
 int data
 struct node * next;
};
struct node * head = NULL;
struct 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, pos;
printf ("In Enter 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_data ();
 insert_end (data);
 break;

case 3:
    printf ("\n Inserting a node at given position");
    data = get_data ();
    position = get_position ();
    insert_mid (position, data);
     break;
Case 4:
    printf ("\n Deleting at beginning");
     delete_begin ();
     break;

case 5:
    printf ("\n Deleting at end \n");
    delete_end;
     break ;
case 6:
    printf ("\n Delete a node at a position\n");
        position= get_position ();
         delete_mid (position); break;}
```

```
}
struct node * create (int data).
{
  struct node * new_node = (struct node *) malloc (sizeof(struct node));
if (new_node == NULL)
  { printf ("\n Can't be allocated \n");
     return NULL;
  }
  new_node -> data = data
  new_node -> next = NULL;
  new_node -> prev = NULL;
   return new_node ;
}
void  insest_begin (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 ;
         retur ;
      }
 else
    head -> prev -> next = new_node;
    new_node -> prev = head -> prev
    new_node -> next = head;
    head -> prev = new_node;
```

```
head = new_node;
}
}
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 -> prev = head
        next
 head -> prev = new_node;
 }
}
void insert_mid (int position, int data)
 { if (position <= 0)
    { printf ("\nInvalid position\n"); }
else if (head == NULL && position >1) {
      printf("\n Invalid Position\n"); }
else if (head != NULL && posito -> large -
else {
   struct node * new_node = create (data);
   ... (position); break; }
```

```
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; }}}
void delete_begin (){
    if (head -> next == head){
        free (head);
        head = NULL
        return; }
struct node * temp = head;
head -> prev -> next = head -> next;
head -> next -> prev = head -> prev;
head -> = head -> next;
free (temp);
temp = NULL; }
void delete_end {
    if (head == NULL){
        printf ("In List is Empty In");
        return; }
else if (head -> next == head){
        free (head);
        head = NULL;
        return; }
```

```
struct node * last_node = head -> prev;
head -> prev = last_node -> prev;
free (last_node);
last_node = NULL;}
void delete_mid ( int position) {
  struct node * temp = head;
  struct node * prev = NULL;
  int i=1;
  while (i< position){
       prev = temp;
       temp= temp -> next;
         i++;}
  prev ->next = temp -> next;
  temp -> next -> prev = prev;
  free (temp);
  temp= NULL;}
  void display (){
  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;
}
```

Advantages and disadvantages of Circular double linked list

Advantages

- Implementation of advanced data structures like Fibonacci heap
- Used with data where we have to navigate front and back
- Circular doubly linked lists are used in multi prossesing
- Can be traversed in both sides

Disadvantages

- Requires additional memory
- More complex than singly linked list
- If not used properly then problem of infinite loop can occur

B.V. Nikhil Reddy
RA2111030010056

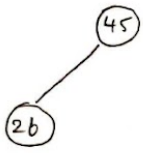
Binary search tree

Assignment-2

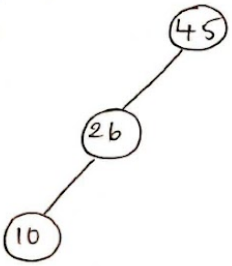
Binary search tree

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

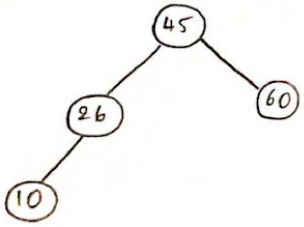
Inserting 26 :



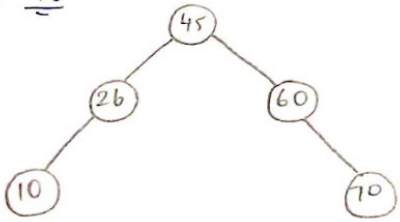
Inserting 10:



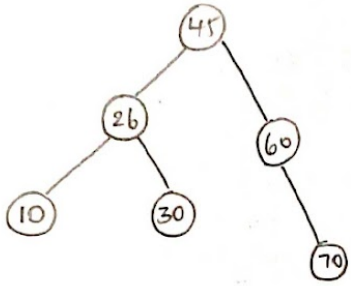
Inserting 60:



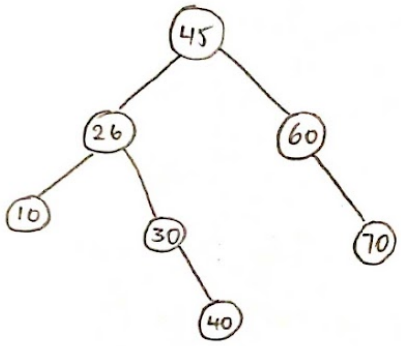
Inserting 70



Inserting 30:

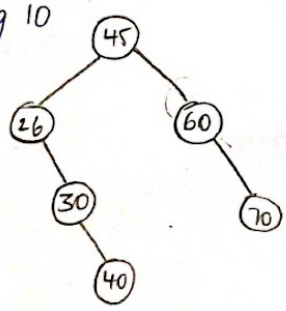


Inserting 40:

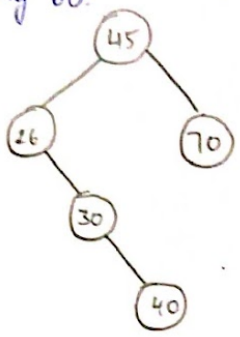


Deleting elements 10, 60, 45

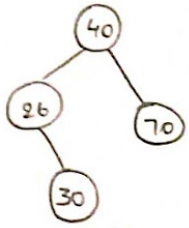
Deleting 10



Deleting 60:



Deleting 45:

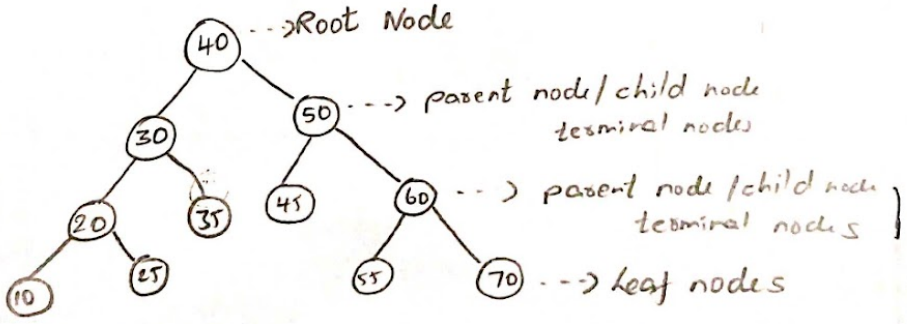


Binary Search Tree

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 I : Start

Step II : Create a new node

Step III : Compare ~~the~~ ~~de~~ If tree is empty make new node as root

Step IV : 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 V : Repeate II until user enters 0

Step VI : End

Pseudo Code

```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int key;
  struct node * left, *right};
  struct newNode (int item) {
  struct node *temp = (struct node *) malloc (sizeof (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 newNode(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(){
    struct node *root = NULL;
    int n=1, data, x;
    ~~sca scanf~~
    while (n != 0)
                    → scanf("%d", &data)
    {root = insert(root, data);}
        ~~scanf~~
        printf("Enter 0 to exit\n");
        scanf("%d", &n);
    } printf("Enter ~~deleted~~ node you want to delete");
        scanf("%d", &x);
    inorder(root);

        root = deleteNode(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 between 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

B.V. Nikhil Reddy
RA2111030010057

Assignment - 3 Hashing

Hash function => $(3x+4) \bmod 7$ starting at index 0

Hash table is initially empty and then the keys are inserted

Given Keys: 1, 3, 8, 10

$h(x) = (3x+4) \bmod 7$

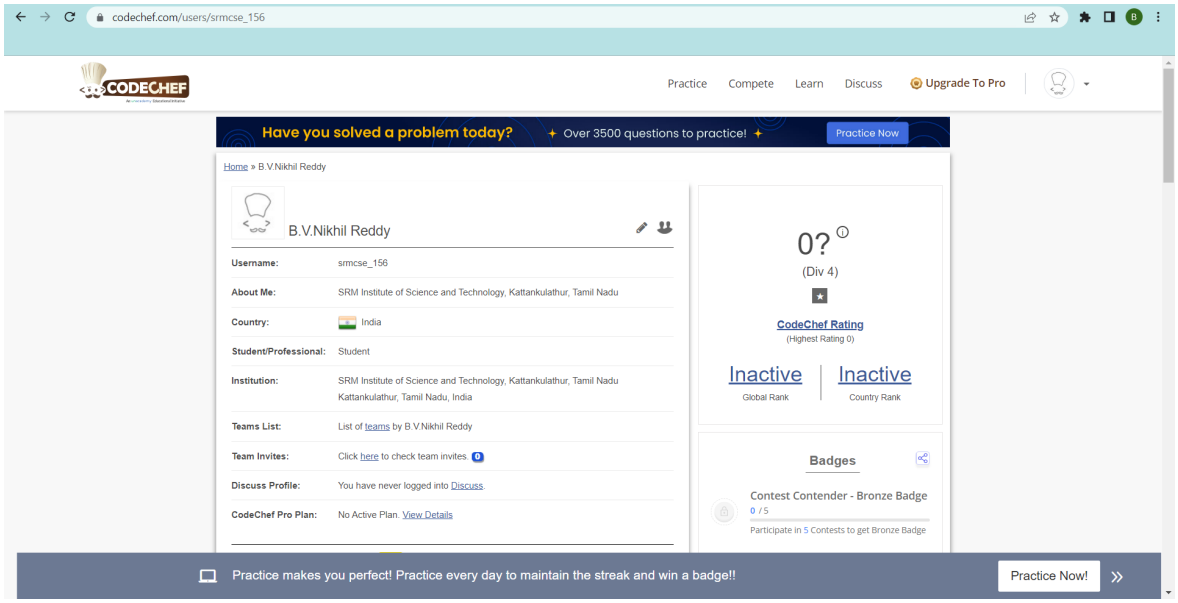
Key	Location
1	$[3(1)+4] \% 7 = 0$
3	$[3(3)+4] \% 7 = 6$
8	$[3(8)+4] \% 7 = 0$ [Put in the next available space] $= 1$
10	$[3(10)+4] \% 7 = 6$ [Put in the next available space] $= 2$

0	1
1	8
2	10
3	
4	
5	
6	3

$\Rightarrow$ 1, 8, 10, –, –, –, 3

Therefore answer is B)

Codechef Achievements



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

Hackerrank profile https://www.hackerrank.com/bb0377

Great Learning

CERTIFICATE OF COMPLETION

Presented to

Nikhil Reddy

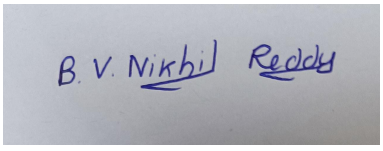
For successfully completing a free online course

Data Structures in C

Provided by

Great Learning Academy

(On November 2022)



Signature

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