| 22/5/2023 | Assignment - 3 | G. Praveen Kumar<br>I'st year AIE -<br>Ch.en.U4aie22014 |
|---|---|---|

① the four operations which can be performed are (x++, ++x, x--, --x)

⇒ Program :-

```
#include<stdio.h>
#include<string.h>
int main(){
    int x=0, length, i;
    Printf("Enter number of operation: ");
    scanf("%d", &length);
    printf("Enter operations to get excuted : \n");
    for(i=0; i<length; i++)
    {
        char operations[4];
        scanf("%s", operations);
        if(strcmp(operations,"x++")==0 || strcmp(operations,"++x")==0)
        {
            x++;
        }
        else if(strcmp(operations,"x--")==0|| strcmp(operations,"--x")==0)
        {
            x--;
        }
    }
    Printf("%d : the value of x", x);
    return 0;
}
```

Input/Output:-

a) Enter number of operations : 3
Enter operation to get executed :
++x
++x
--x
1 : the value of x

b) Enter number of operations : 4
Enter operation to get executed :
++x
x++
--x
x--
0 : the value of x

---

2)

Program:-

```
#include <stdio.h>
int main()
{
    int size;
    printf("Enter number of elements : ");
    scanf("%d", &size);
    int arr[size*2];
    Printf("Enter the elements : \n");
    int i;
    for(i=0; i<size*2; i++)
    {
        scanf("%d", &arr[i]);
    }
```

```
    for(i=0;i<size;i++)
    {
        printf("%d %d ", arr[i], arr[size+i]);
    }
    return 0;
}
```

Output:

a) Enter number of elements : 2
Enter the elements :
1
1
2
2

1 2 1 2

b) Enter number of elements : 4
Enter the elements :
1
2
3
4
4
3
2
1

1 4 2 3 3 2 4 1

(3) A binary array is an array in which each element is either 0 or 1.

a) nums = {1, 0, 1, 1, 0, 1}

b) nums = {1, 0, 0, 1}

Program:

```
#include <stdio.h>
int main()
{
    int size, max=0, count=0, i;
    Printf("Enter number of elements: ");
    Scanf("%d", &size);
    int arr[size];
    Printf("Enter the numbers: ");
    for(i=0; i<size; i++)
    {
        Scanf("%d", &arr[i]);
    }
    for(i=0; i<size; i++)
    {
        if(arr[i]==1)
        {
            count+=1;
        }
        else
        {
            count=0;
        }
        if(count > max)
        {
            max= count;
        }
    }
```

```
        Printf("Maximum number of consecutive: %d", maz);
        return 0;
}
```

Output:

a) Enter the number of Elements: 6

Enter the numbers:

1
0
1
1
0
1

Maximum number of Consecutive: 2

b) Enter the number of Elements: 4

Enter the numbers:

1
0
0
1

Maximum number of Consecutive: 1

---

4) A subarray is a contigous non-empty sequence of elements within an array.

a) nums = {1, 3, 0, 0, 2, 0, 0, 4}

b) nums = {2, 10, 2019}

Program:

```
#include<stdio.h>
int count (int num[], int size)
{
    int count=0;
```

```
for(int i=0; i<size;i++)
{
    if(num[i]==0)
    {
        count++;
    }

int j;
  j=i+1;
  while(j<size && num[j]==0)
  {
      Count++;
      j++;
  }
 }
}
 return count;
}

int main()
{
    int size;
    scanf("%d", &size);
    int num[size];
    for(int i=0;i<size;i++)
    {
        scanf("%d",&num[i]);
    }
    int zero;
      zero=count(num,size);
```

```
    Printf("Result : %d", result);
    return 0;
}
```

Output:

a) size : 8
1
3
0
0
2
0
0
4
Result : 6

b) size : 3
2
10
2019
Result : 0

---

5) Linked List

```
#include<stdio.h>
struct ListNode {
    int data;
    struct ListNode *next;
};
Void create(int num, struct ListNode *head)
{
```

```
struct ListNode *temp = head;
struct ListNode *temp = (struct ListNode*)malloc(sizeof(struct
);
while(temphead->link != NULL)
{
    temphead = temphead->link;
}
temphead->link = temp;
temp->data = num;
temp->link = NULL;
}

Void main()
{
    Struct ListNode *head = (struct node*)malloc(sizeof(struct node));
    Create(2, head);
    Create(1, head);
    Struct ListNode *fp = head;
    Struct ListNode *sp = head;
    while(fp != NULL && fp->next != NULL)
    {
        sp = sp->next;
        fp = fp->next->next;
    }
    Printf("Middle Node = %d", sp->data);
}
```

OUTPUT: 1→3→4→7→1→2→6→NULL

a) Middle Node = 7

b) Middle Node = 1

2→1→NULL

6) An input string is valid if:

Program:

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

struct stack {
        int top;
        unsigned cap;
        char* array;
};

struct stack* create(unsigned cap)
{
        struct stack* stack = (struct stack*) malloc(sizeof
                                                (struct stack));
        stack->cap = cap;
        stack->top = -1;
        stack->array = (char*) malloc(stack->cap * sizeof(char));
        return stack;
}

bool isEmpty(struct stack* stack)
{
        return stack->top == -1;
}

void push(struct stack* stack, char item)
{
        stack->array[++stack->top] = item;
}
```

```
char pop(struct stack * stack)
{
    if (isEmpty(stack))
    {
        Printf("stack is Empty");
        return '\0';
    }
    return stack -> array[stack->top--];
}
bool match(char character1, char character2)
{
    if(character1 == '(' && character2 == ')')
        return true;
    if(character1 == '{' && character2 == '}')
        return true;
    if(character1 == '[' && character2 == ']')
        retur true;
    return false;
}
bool valid(char *s){
    struct stack * stack = create(strlen(s));
    for(int i=0; i< strlen(s); i++)
    {
        if(s[i] == '(' || s[i] == '{' || s[i] == '[')
            push(stack, s[i]);
        else if(s[i] == ')' || s[i] == '}' || s[i] == ']')
            if(isEmpty(stack))
                return false;
```

```
        }
    }
    return isEmpty(stack);
}

int main()
{
    char s[100];
    scanf("%s", &s);
    bool isValid = valid(s);
    if (isValid)
        printf("true\n");
    else
        printf("false\n");
    return 0;
}
```

Output:

a) "[]"

false

b) "({[]})"

true

(3)

Program :

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

struct node{

    int data;
    struct node *link;
};

Void create (struct node *head, int n)
{
    struct node *temp = head;
    for(int i=0; i<n; i++)
    {
        if(i!=n-1)
        {
            scanf("%d", &temp->data);
            temp->link=(struct node*)malloc(sizeof(struct node));
            temp= temp->link;
        }
        else
        {
            scanf("%d", &temp->data);
            temp->link = NULL;
        }
    }
}
```

```
int valid ( struct node * rear , struct node * temp , int n )
{
        int count =0 ;
        for (int i= 0; i<n ; i++)
        {
            if (rear -> data! =0 && temp -> data! =0)
            {
                rear -> data = rear -> data - 1 ;
                rear= rear -> link;
                count += 1 ;
            }
            else {
              rear= rear -> link;
            }
        }
    }
    return count;
}
void find(int k, struct node * head , int n)
{
        struct node * start , * temp=head, *rear;
        start = head;
        rear = start ;
        int check = 1, count=0 ;
        for( int i=0; i<k ; i++)
        {
            temp = temp -> link ;
        }
```

```
    while(check==1)
    {
        if(temp->data!=0)
        {
            count+=valdel(rear,temp,n);
            rear=start;
        }
        else
        {
            check=0;
        }
    }
    printf("tim taken to by ticket:%d s", count);
}
int main()
{
    int n,k, count=0;
    printf("enter no of persons:");
    scanf("%d",&n);
    printf("Enter required ticke to buy for each person: \n");
    struct node *head=(struct node*) malloc(sizeof(struct node));
    create(head,n);
    printf("Enter person you want to check time for:");
    scanf("%d",&k);
    find(k,head,n);
}
```

Output:

Enter number of Person: 3

Enter required tickets to buy for each person: 2 3 2

Enter the person you want to check time for: 2

time taken to buy ticket: 6s

3) a) tickets = {2, 3, 2}, k = 2

Program:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node* NODE;
struct Node {
        NODE left;
        NODE right;
        int data;
};
NODE create(int val)
{
        NODE n = (NODE) malloc (sizeof (struct Node*));
        n -> data = val;
        n -> left = NULL;
        n -> right = NULL;
        return n;
}
Void preorder(NODE node)
{
        if (node == NULL)
        {
            return;
        }
        printf("%d", node -> data);
        Prooorder (node -> left);
        preoorder (node -> right);
}
```

```
Void insert ( NODE tree, int val )
{
    if( val == tree → data)
    {
        return ;
    }
    if (val < tree → data)
    {
        if(tree → left == NULL)
        {
            tree → left = = create(val);
            return;
        }
        else
        {
            return insert (tree → left, val);
        }
    }
    if (val > tree → data)
    {
        if(tree → right = = NULL)
        {
            tree → right = create (val);
            return;
        }
        else
        {
            return insert(tree→right, val);
        }
    }
}
```

```
int main ()
{
    int arr[] = { 1, 2, 5, 3, 6, 4 };
    int len = 6;
    NODE Root = Create (arr[0]);
    for( int i = 0; i < len; i++)
    {
    }     insert(Root, arr[i]);
    Preorder (Root);
    return 0;
}
```

output:

Enter number of nodes: 15

Enter the value of nods:

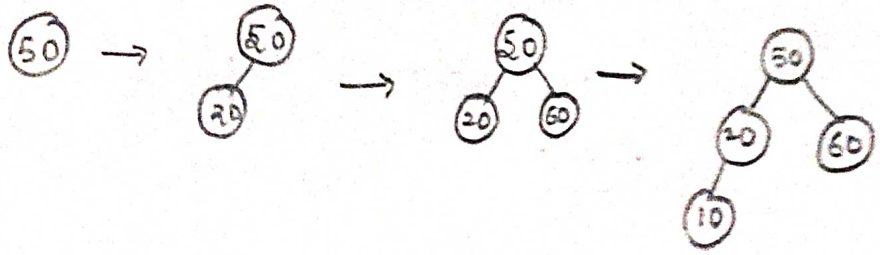
1
14
3
7
4
5
15
6
13
10
11
2
12
8
9

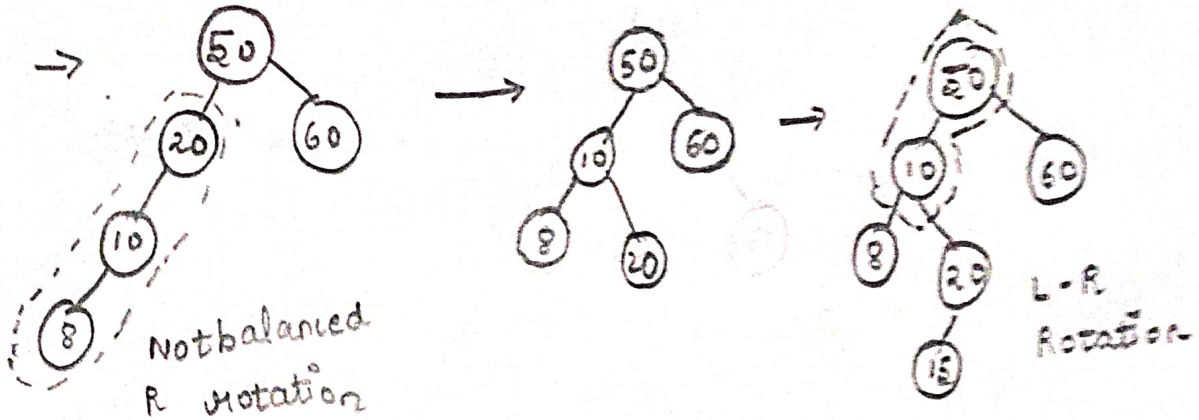
Pre-order traversal:

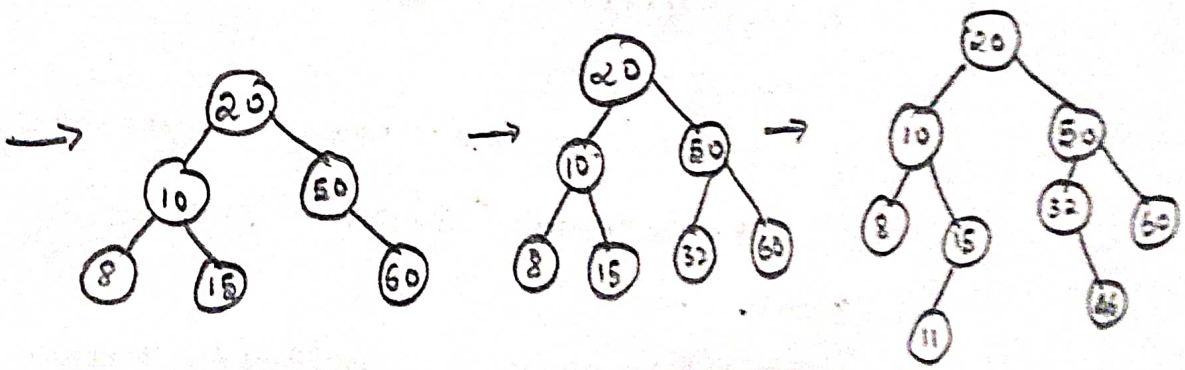
1 14 3 2 7 4 5 6 13 10 8 9 11 12 15

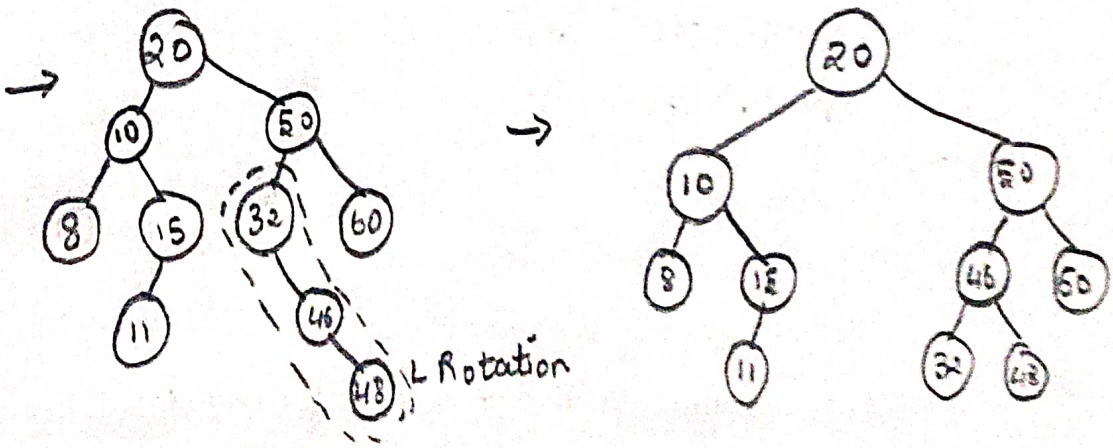
9)

50, 20, 60, 10, 8, 15, 32, 46, 11, 48



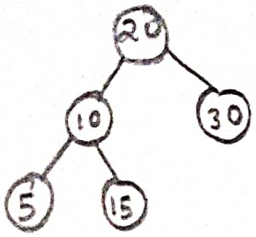



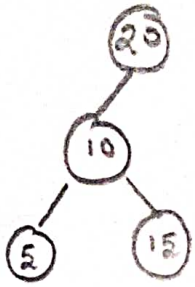



10)

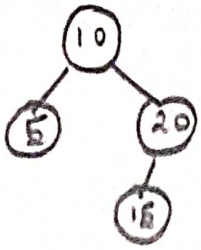
a) Delete node 30



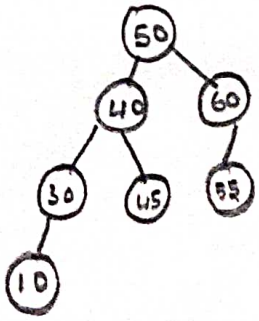

↓ delete 30



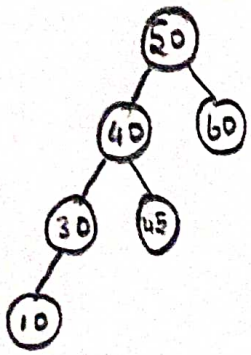

balance → right-rotation




b) Delete node 55




delete 55




balance → right-rotation