**Section (I): Tracing Questions (Total: 8 marks)**

**(Q1) (3 mark)**

* Draw the binary expression tree for the following prefix expression.
* Also, write the corresponding postfix, and infix expression.
* **\* \* + A – B K D \* \* C W E**

**(Q2) (5 mark)**

**For exercises (i - iii) use the following values, and draw the hash table for each of them:**

**24 , 26 , 38, 28 , 200 , 31 , 19 , 42 , 22**

1. Store the values in a hash table with size 10 cells.
2. Store the values in a hash table with 4 buckets, each bucket contains 3 cells.
3. Store the values in a hash chain table with function (Key % 4).
4. Fill in the following table, showing the number of comparisons needed to find each value:-

|  |  |  |  |
| --- | --- | --- | --- |
|  | **(i)** | **(ii)** | **(iii)** |
| **19** |  |  |  |
| **22** |  |  |  |
| **42** |  |  |  |

**Section (II): Algorithm Questions (Total: 8 marks)**

**Algorithm 1: (8 marks)**

You are asked to write a function called **Remove\_Two\_ Leafs** **(** **int** **V** **)**

* Your function will receive an integer that represents the some value.
* You will find some node which owns 2 leafs with the total of those leafs equals to (**V**).

**e.g. V = 232**

e.g. if the required V = 232

then : your function will remove those 2 leafs

because : (112 + 120) = 232

**112**

**120**

115

**90**

**70**

80

50

**160**

**600**

110

500

122

155

130

void TotalLeafs(CTNode\* pT, CTNode\* pB, CTNode\*&pL1,CTNode\*&pL2, CTNode\*& pb1, CTNode\*& pb2, int& ct, int& tot, int v)

{

if (pT == NULL)

return;

TotalLeafs(pT->pLeft, pT, pL1, pL2, pb1, pb2, ct, tot, v);

if (pT->pLeft == NULL && pT->pRight == NULL)

{

ct++;

if (ct == 1)

{

pb1 = pB;

pL1 = pT;

}

if (ct == 2)

{

pb2 = pB;

pL2 = pT;

if (pb1 == pb2)

{

tot = pL1->info + pL2->info;

}

if (v == tot)

{

pb1->pLeft = pb1->pRight = NULL;

delete pL1;

delete pL2;

}

ct = 0;

tot = 0;

pb1 = pb2 = pL1 = pL2 = NULL;

}

}

TotalLeafs(pT->pRight, pT, pL1, pL2, pb1, pb2, ct, tot, v);

}

void Remove\_Two\_Leafs(int v)

{

CTNode\* pT = pRoot;

int v;

cin >> v;

CTNode\* pL1, \* pL2, \* pb1, \* pb2 = NULL;

int ct = 0, tot = 0;

TotalLeafs(pT, NULL, pL1, pL2, pb1, pb2, ct, tot, v);

}

**Section (III): Problem Solving (Total: 24 marks)**

**Problem 1: (5 marks)**

*In the this question use the following definition for the nodes in the List:*

**class CListNode**

**{**

**public:**

info

pNext

**int info;**

**CListNode \*pNext;**

**};**

Write a **recursive** function to do:

Display the 2nd and the 3rd quarters of the list in reverse manner.

**Assume:** that the length (number of nodes = 8 in the example) of the list is given to your function as a parameter.

Also

**Assume :** that the length is divisible by 4.

3rd quarter

2nd quarter

Head

**The output :**

**130 , 73 , 17 , 44**

void ReverseDisp(CNode\* pT, CNode\* pB, int N, int& check, int&ct)

{

if (pT == NULL)

return;

ct++;

if (ct == ((N / 4) + 1) || ct == (N - N / 4))

{

check++;

if (check == 2)

{

check--;

ct--;

cout << pT->info;

return;

}

}

ReverseDisp(pT->pNext, pT, N, check, ct);

if (ct == (N / 4) || ct == (N - N / 4))

{

check--;

ct--;

}

if (check != 0)

{

cout << pT->info;

}

}

**Problem 2: (19 marks)**

Write a main function to do the following:

1

4

3

2

1. Read a Binary Tree (**T1**).
2. Ask the user to enter target value (**V**).
3. Find (**V**),

if the last node of one of the 2 branchs not a leaf,

then cut its sub-tree and past it to a new list (**Lst**).

e.g.

60

655

660

85

**75**

70

80

510

90

920

100

1530

1390

1350

1330

1500

1400

1340

V = 4000

1550

1540

1700

1545

4000

1600

3000

8200

5100

2900

9500

5050

5800

2000

**9800**

5900

5300

**9900**

2850

6500

5400

5200

2100

6000

2050

2200

2300

2250

2400

Lst

Head

**2400**

2300

2200

2850

**2050**

**2250**

2100

1. Copy the nodes of the 2 branches to new Tree, and add it to the (**Lst**).

2850

Lst

Head

**2050**

**2250**

**2400**

2300

2200

4

3000

2100

8200

2900

9500

2000

**9800**

**9900**

2

1. Repeate step-2 & step-3 & step-4

e.g.

V = 100

.

9900

1545

1600

1700

1540

1550

1390

1350

1340

1400

2000

1500

1530

6000

9800

5050

5400

6500

5900

5300

5800

5200

2900

9500

5100

3000

8200

4000

655

660

85

**75**

70

80

1330

510

90

920

100

60

3

1. Display all trees in the (**Lst**).

#include <iostream>

using namespace std;

class CTNode{

public:

int info;

CTNode\* pLeft;

CTNode\* pRight;

};

class CNode

{

public:

int info;

CTNode\* pDown;

CNode\* pNext;

};

int main()

{

CBST tree, subTree;

CList Lst;

int n;

cin>>n;

while (n--)

{

CTNode\* p=new CTNode;

cin>>p->info;

p->pLeft = p->pRight =NULL;

tree.Insert(p);

}

int checkSubTree = 1;

while (checkSubTree)

{

int v ;

cin>>v;

CTNode\* pTrav = tree.pRoot;

while (pTrav != NULL && pTrav->info != v)

{

if(v>pTrav->info)

{

pTrav = pTrav->pRight;

}else{

pTrav = pTrav->pLeft;

}

}

// check right

CTNode\* pSideRight = pTrav;

while (pSideRight->pRight != NULL)

{

pSideRight = pSideRight->pRight;

}

// check left

CTNode\* pSideLeft = pTrav;

while (pSideLeft->pLeft != NULL)

{

pSideLeft = pSideLeft->pLeft;

}

if(pSideRight->pLeft == NULL && pSideLeft->pRight == NULL)

{

checkSubTree=0;

break;

}

if(pSideRight->pLeft != NULL)

{

CNode\* pnn = new CNode;

pnn->pDown = pSideRight->pLeft;

pnn->pNext = NULL;

Lst.Attach(pnn);

pSideRight->pLeft = NULL;

}

if(pSideLeft->pRight != NULL)

{

CNode\* pnn = new CNode;

pnn->pDown = pSideLeft->pRight;

pnn->pNext = NULL;

Lst.Attach(pnn);

pSideLeft->pRight = NULL;

}

pSideLeft = pTrav->pLeft;

subTree.pRoot =NULL;

while (pSideLeft->pLeft != NULL)

{

CTNode\* p=new CTNode;

p->info = pSideLeft->info;

p->pLeft = p->pRight =NULL;

subTree.Insert(p);

pSideLeft = pSideLeft->pLeft;

}

pSideRight = pTrav->pRight;

while (pSideRight->pRight != NULL)

{

CTNode\* p=new CTNode;

p->info = pSideRight->info;

p->pLeft = p->pRight =NULL;

subTree.Insert(p);

pSideRight = pSideRight->pRight;

}

CNode\* pnn = new CNode;

pnn->pDown = subTree.pRoot;

pnn->pNext = NULL;

Lst.Attach(pnn);

}

}