**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 + C \* D – E F G + H – K – L M**

**(Q2) (5 mark)**

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

**62 , 81 , 63 , 64 , 86 , 66 , 46 , 39**

1. Store the values in a hash table with size 8 cells.
2. Store the values in a hash table with 3 buckets, each bucket contains 3 cells.
3. Store the values in a hash chain table with function (Key % 8).
4. Fill in the following table, showing the number of comparisons needed to find each value:-

|  |  |  |  |
| --- | --- | --- | --- |
|  | **(i)** | **(ii)** | **(iii)** |
| **86** |  |  |  |
| **46** |  |  |  |
| **39** |  |  |  |

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

**Algorithm 2: (8 marks)**

You are asked to write a function called **Display\_Those** **()**

a) Find the node of the **Largest** value, and display it.

b) If the largest node, has a child, then find the **Smallest** node of **its sub-tree,** and display it**.**

c) from the node you reached, Repeat (a) & (b) as much as you found nodes.

760

625

627

628

875

705

720

710

750

850

630

620

870

885

880

80

50

112

120

115

160

**900**

110

500

122

155

130

The output will be:

900

600

850

700

720

775

770

**720**

**700**

70

610

**600**

void Display\_Those()

{

CTNode\* pTrav = pRoot;

while (1)

{

while (pTrav->pRight != NULL)

{

pTrav = pTrav->pRight;

}

cout << pTrav << endl;

while (pTrav->pLeft != NULL)

{

pTrav = pTrav->pLeft;

}

cout << pTrav << endl;

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

{

break;

}

}

}

**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;**

**};**

**Note:**

* in this problem you have to write just single function

(**no permission** to write more than 1 function).

* **no permission** to use loop inside the function.

Write a **recursive** function to do the following:

**Assume:** that the lengths of the 2 lists are equal.

Your function should :

* display the nodes in (L1) , but in reverse manner.
* display the nodes in (L2) , but in forward manner.
* When you face (-1) switch to the another List.

e.g.

**outputs** : **90 , 75 , 40, 33 , 38 , 14 , 94**  **:** 50, 11, 44, 15, 39, 90, 60, 43, 80, 70

L1

Head

L2

Head

void DispSpecial(CNode\* pT1, CNode\* pT2, int f)

{

if (pT1 == NULL)

return;

if (pT1->info == -1 || pT2->info == -1)

{

if (pT1->info == -1)

{

f = 1;

cout << pT2->info;

DispSpecial(pT1->pNext, pT2->pNext, f);

}

else

{

f = 0;

DispSpecial(pT1->pNext, pT2->pNext, f);

cout << pT1->info;

}

}

else

{

if (f == 0)

{

DispSpecial(pT1->pNext, pT2->pNext, f);

cout << pT1->info;

}

else

{

cout << pT2->info;

DispSpecial(pT1->pNext, pT2->pNext, f);

}

}

}

**Problem 2: (19 marks)**

Write a main function to do the following:

1. Read the required data structure **(L)** according to the following protocol:

95

80

90

20

70

130

100

90

95

120

81

94

82

83

80

115

230

240

250

310

300

80

200

1200

950

940

1300

330

320

88

**L**

Head

280

87

85

82

85

* + Ask the user to determine the number of rows he needs to enter (**nR**). 🡪 in the example nR = 5
  + For the1st rowread just 1 node, for the 2nd row read 2 nodes , and so on.
  + **In the last row**: you have to read some binary search tree from the user for each node, and points to it by down pointer of the node.

**Note:**

You are responsible to declare the data structure for the ListNode.

1. Ask the user to select a column (iC), and then display all trees starting from (iC) but only the 1st 3 nodes starting from the required node.

82

**85**

95

80

90

330

230

320

240

250

310

300

80

**200**

88

95

90

100

80

115

81

94

82

83

130

120

20

70

940

1300

1200

950

280

87

85

e.g. ( iC = 4 🡪 [ 82, 85 ] 🡪 [ 80 , 200 , 300] )

**L**

Head

Another example for step(2)

e.g. ( iC = 3 🡪 [ 90, 100 , 120 ] 🡪 [ 80 , 85 , 90] 🡪 [230 , 240] )

280

87

85

82

85

95

80

**90**

330

230

320

**240**

250

310

300

80

**200**

88

95

90

100

80

115

81

94

82

83

**L**

130

**120**

20

70

940

1300

1200

950

Head



class CNode

{

public:

int info;

CNode\* pNext;

CNode\* pDownL;

CTNode\* pDownT;

};

void Visit(CTNode\* pp, int& ct)

{

if (pp == NULL || ct == 3)

return;

ct++;

Visit(pp->pLeft, ct);

cout << pp->info << " ";

Visit(pp->pRight, ct);

}

void main()

{

CList L;

CBST T;

CNode\* pmyTail = NULL, \* pCurrent = NULL, \* pLast = NULL, \*pnn=NULL;

int nr;

cin >> nr;

for (int i = 1; i <= nr; i++)

{

pmyTail = NULL;

pCurrent = pLast;

for (int j = 0; j < i; j++)

{

pnn = new CNode;

cin >> pnn->info;

pnn->pNext = NULL;

pnn->pDownL = NULL;

pnn->pDownT = NULL;

if (i == 1)

{

L.pHead = pnn;

}

else

{

if (pmyTail == NULL)

{

pmyTail = pnn;

}

else

{

pmyTail->pNext = pnn;

pmyTail = pnn;

}

if (pCurrent != NULL)

{

pCurrent->pDownL = pnn;

pCurrent = pCurrent->pNext;

}

}

}

if (i == 1)

{

pLast = pnn;

}

else

{

pLast = pLast->pDownL;

}

}

int N;

while (pLast != NULL)

{

cin >> N;

for (int i = 0; i < N; i++)

{

CTNode\* pnT = new CTNode;

cin >> pnT->info;

pnT->pLeft = NULL;

pnT->pRight = NULL;

T.Insert(pnT);

}

pLast->pDownT = T.pRoot;

T.pRoot = NULL;

pLast = pLast->pNext;

}

int iC,ct=0;

cin >> iC;

CNode\* pT = L.pHead;

for (int i = 0; i < (iC - 1); i++)

{

pT = pT->pDownL;

}

for (int i = 0; i < (iC - 1); i++)

{

pT = pT->pNext;

}

CNode\* pK = pT;

while (pK->pDownL != NULL)

{

pK = pK->pDownL;

}

while (pT != NULL)

{

CTNode\* pp = pK->pDownT;

while (pp != NULL && pp->info != pT->info)

{

if (pT->info > pp->info)

{

pp = pp->pRight;

}

else

{

pp = pp->pLeft;

}

}

ct = 0;

Visit(pp, ct);

pT = pT->pDownL;

pK = pK->pNext;

}

}