Prob 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | 21 |  |  |  |  |  |  |  |
|  |  |  | 15 |  |  |  |  |  |  |  | 45 |  |  |  |
|  | 7 |  |  |  |  |  |  |  | 30 |  |  |  | 65 |  |
| 5 |  |  |  |  |  |  |  | 25 |  |  |  |  |  |  |

(1)

(2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 7 |  |  |
|  |  | 15 |  |  |  |  |  |  | 65 |
|  |  |  |  | 25 |  |  |  |  |  |
|  |  |  | 21 |  | 30 |  |  |  |  |
|  |  |  |  |  |  | 45 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Prob 2.

typedef struct treeNode {

int data;

struct treeNode\* left;

struct treeNode\* right;

} treeNodeType;

int TreeSize(treeNodeType\* T)

{

int count = 1;

count += TreeSize(t->left) + TreeSize(t->right)

return count;

}

Prob 3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | J |  |  |  |
|  |  | I | F |  | G | H | H |  |  |
| A | B | C | C | E | D | D | D | D |  |

Prob 4.

(1)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **A** | 0 |  | 1 |  |  |  | 1 |  |  |
| **B** |  | 0 |  | 1 |  |  |  |  |  |
| **C** | 1 |  | 0 |  | 1 |  |  |  |  |
| **D** |  | 1 |  | 0 | 1 | 1 |  |  |  |
| **E** |  |  | 1 | 1 | 0 |  |  |  |  |
| **F** |  |  |  | 1 |  | 0 |  |  |  |
| **G** | 1 |  |  |  |  |  | 0 | 1 |  |
| **H** |  |  |  |  |  |  | 1 | 0 | 1 |
| **I** |  |  |  |  |  |  |  | 1 | 0 |

(2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | C | G |  |  |
| **B** | D |  |  |  |
| **C** | A | D | E |  |
| **D** | B | C | E | F |
| **E** | C | D |  |  |
| **F** | D |  |  |  |
| **G** | A | H |  |  |
| **H** | G | I |  |  |
| **I** | H |  |  |  |

Prob 5.

(1) C-E(1), E-I(2) , A-C(2), D-E(3), D-F(3), A-G(3), C-H(4), A-B(5)

(2) A-C(2), C-E(1), E-I(2), E-D(3), D-F(3), C-H(4), A-G(3), A-B(5)

(3) A-B(5), A-C(2), C-D(5), D-F(3), C-E(1), E-I(2), A-G(3), C-H(4)

Prob 6.

[**0**, **5**, INF, **7**, INF, INF, **3**, INF, INF]

[0, 5, INF, 7, INF, INF, 3, 1**0**, INF]

[0, 5, INF, 7, INF, INF, 3, 10, INF]

[0, 5, INF, 7, **10**, **10**, 3, 10, INF]

[0, 5, INF, 7, 10, 10, 3, 10, INF]

[0, 5, **14**, 7, 10, 10, 3, 10, **12**]

Prob 7.

G A D B C E F

Prob 8.

typedef struct treeNode {

int data;

struct treeNode\* left;

struct treeNode\* right;

} treeNodeType;

treeNodeType\* T = NULL;

void DescendingOrder(treeNodeType\* T)

{

if (T != null) {

DescendingOrder(T->right);

printf(“%d ”, T->data);

DescendingOrder(T->left);

}

}

treeNodeType\* indert(treeNodeType\* T, int data) {

if (T == NULL) {

T = (treeNodeType\*)malloc(sizeof(treeNodeType));

T->left = T->right = NULL;

T->data = data;

} else {

if (T->data > data)

T->left = insert(T->left, data)

else

T->right = insert(T->right, data)

}

}

void main()

{

while(data > 0) {

int data;

scanf(“%d”, &data);

insert(T, data);

}

DescendingOrder(T);

}