1. (1)

struct Person {

char name[10];

int age;

float salary;

};

1. (2)

if(strcmp(p1->name, p2->name)) return 0;

if(p1->age != p2->age) return 0;

if(p1->salary != p2->salary) return 0;

return 1;

1. (3)

if(humans\_equal(&p1, &p2))

2. (1)

rv = (char\*)malloc(sizeof(char)\*(strlen(dst) + strlen(src) + 1));

2. (2)

for(i=0; j<strlen(src); i++) {

rv[i + strlen(dst)] = src[i];

}

rv[strlen(dst) + strlen(srv)] = ‘\0’;

3. (1)

stack[++top] = item;

3. (2)

return stack[top--];

3. (3)

0 inserted.

10 inserted.

20 inserted.

top: 2

20 deleted.

10 deleted.

0 deleted.

top: -1

4. (1)

for(i=front + 1; i < MAX\_QUEUE\_SIZE; i++) {

queue[i - (front + 1)] = queue[i];

}

4. (2)

queue[++rear] = item;

4. (3)

return queue[++front];

4. (4)

front: -1 rear: -1

front: -1 rear: 7

front: 7 rear: 7

front: -1 rear: 7

front: 7 rear: 7

5. (1)

data[i] = rand() % k + 1;

5. (2)

if(data[j] == data[i]) break;

5. (3)

if(j < i) i--;

6. (1)

if(currNode->key == key) return;

if(currNode->link == NULL) break;

currNode = currNode->link;

6. (2)

if(currNode->key == key) break;

prevNode = currNode;

currNode = currNode->link;

6. (3)

while(currNode) {

printf(”%d “, currNode->key);

currNode = currNode->link;

}

printf(”\n”);

6. (4)

while(first) {

temp = first;

first = first->link;

free(temp);

}

7. (1)

if(ptr) {

recursive\_inorder(ptr->left\_child);

printf(”%c ”, ptr->data);

recursive\_inorder(ptr->right\_child);

}

7.(2)

if(ptr) {

recursive\_postorder(prt->left\_child);

recursive\_postorder(ptr->right\_child);

printf(”%c “, ptr->data);

}

8. (1)

13

8. (2)

3. 한 노드의 최대 자식 노드 수가 3이기 때문.

8. (3)

No. D has 3 childs.

8. (4)

마지막 level을 제외한 level에 node가 가득 차 있고, 마지막 level에는 node가 왼쪽부터 채워지는 형태

9.

O(2^n), O(n^3), O(n^2), O(n log n), O(n), O(log n), O(1)

10. (1)

O(n^2)

10. (2)

O(n^4)

10. (3)

O(n log n)

10. (4)

O(2^n)