1번

#include <stdlib.h>

#include <stdio.h>

int oneOrTwo(int num) { //일의자리 수인지 십의 자리 수인지 계산하여 count10에 곱해야할 10의 배수 결정

if(num/10 == 0) return 10;

else return 100;

}

void makeS(int A[], int S[]) {

int index = 0;

int count10 = 1;

int value = 0;

int a,b,c,d,e;

for(a=0; a<5; a++) {

value += A[a]\*count10;

count10 \*= oneOrTwo(A[a]);

printf("value = %d\n", value);

for(b=0; a!=b && b<5; b++) {

value += A[b]\*count10;

count10 \*= oneOrTwo(A[b]);

printf("value = %d\n", value);

for(c=0; c!=a && c!=b && c<5; c++) {

value += A[c]\*count10;

count10 \*= oneOrTwo(A[c]);

printf("value = %d\n", value);

for(d=0; d!=a && d!=b && d!=c && d<5; d++) {

value += A[d]\*count10;

count10 \*= oneOrTwo(A[d]);

printf("value = %d\n", value);

for(e=0; e!=a && e!=b && e!=c && e!=d && e<5; e++) {

value += A[e]\*count10;

count10 \*= oneOrTwo(A[e]);

printf("value = %d\n", value);

S[index] = value;

index++;

}

}

}

}

}

}

void insertSort(int S[]) {

int key;

int j,i;

for(i=1; i<120; i++) {

key = S[i];

while (j>=0 && S[j]<key)

{

S[j+1] = S[j];

j--;

}

S[j+1] = key;

}

}

int main() {

int A[5] = {6,4,41,8,44};

int S[120];

makeS(A, S);

for(int i=0; i<120; i++) {

printf("(%d) ", S[i]);

}

printf("\n");

insertSort(S);

}

2번

#include <stdlib.h>

#include <stdio.h>

typedef struct TreeNode

{

int key;

struct TreeNode \*left;

struct TreeNode \*right;

}TreeNode;

TreeNode \*makeNode(int key) {

TreeNode \*node = (TreeNode\*)malloc(sizeof(TreeNode));

node->left = NULL;

node->right = NULL;

node->key = key;

return node;

}

TreeNode \*insert(TreeNode \*root, int key) {

if(root==NULL) return makeNode(key);

else if(root->key < key) root->right = insert(root->right, key);

else if(root->key > key) root->left = insert(root->left, key);

return root;

}

typedef struct QueueType

{

int front, rear;

TreeNode \*data[100];

}QueueType;

void init(QueueType \*Q) {

Q->front = 0;

Q->rear = 0;

}

int isEmpty(QueueType \*Q) {

if(Q->front==Q->rear) return 1;

else return 0;

}

void inqueue(QueueType \*Q, TreeNode \*node) {

Q->data[Q->rear++] = node;

}

TreeNode \*dequeue(QueueType \*Q) {

TreeNode \*node = Q->data[Q->front];

Q->front++;

return node;

}

void bfs(TreeNode \*root) {

QueueType Q;

init(&Q);

TreeNode \*p;

inqueue(&Q, root);

while (!isEmpty(&Q))

{

p = dequeue(&Q);

printf("%d ", p->key);

if(p->left != NULL) inqueue(&Q, p->left);

if(p->right != NULL) inqueue(&Q, p->right);

}

}

int main() {

TreeNode \*root = NULL;

root = insert(root, 5);

root = insert(root, 2); root = insert(root, 1); root = insert(root, 3); root = insert(root, 4);

root = insert(root, 8); root = insert(root, 6); root = insert(root, 9); root = insert(root, 7);

root = insert(root, 10);

bfs(root);

}

