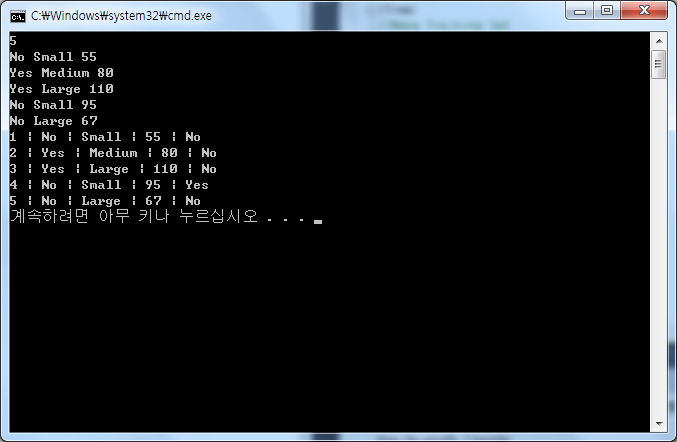
**Training Set**

*201133216*

*정유석*

**<PRINT>**



**<CODE>**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

//Make tree

typedef struct tree{

char attribute[10];

int check,cnt;

struct tree \*right;

}Tree;

//Make Training Set

typedef struct{

int Tid;

char Attrib1[10];

char size[10];

char Class[10];

int cost;

}Training;

//Create set

Training set[30];

//Create Tree

Tree\* root;

Tree\* check1;

Tree\* check2;

//Calculate class

void calculate(Tree\* ,int );

//initialize

void initialize(){

//Set standard

root = (Tree \*)malloc(sizeof(Tree));

strcpy(root->attribute,"Yes");

root->cnt = 0; //Set height

check1 = (Tree \*)malloc(sizeof(Tree));

root->right = check1;

strcpy(check1->attribute,"Medium");

check1->cnt = 1;

check2 = (Tree \*)malloc(sizeof(Tree));

check1->right = check2;

check2->check = 85;

check2->cnt = 2;

}

//Main function

void main(){

int i,N=1000;

char a[50];

initialize();

//Receive number of test set

while(N>=30)

scanf("%d",&N);

//Receive data

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

scanf("%s %s %d",&set[i].Attrib1,&set[i].size,&set[i].cost);

fflush(stdin);

set[i].Tid = i+1; //Set ID

calculate(root,i); //calculate set

}

//Display

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

printf("%d | %s | %s | %d | %s \n",set[i].Tid,set[i].Attrib1,set[i].size,set[i].cost,set[i].Class);

}

//Calculate function

void calculate(Tree\* checkClass, int N){

//Tree height = 0 ==> root

if(checkClass->cnt==0){

if(strcmp(set[N].Attrib1, checkClass->attribute)==0)

strcpy(set[N].Class,"No"); //set Class

else

calculate(checkClass->right,N);

}

//Tree height = 1 ==> root->right

if(checkClass->cnt==1){

if(strcmp(set[N].size, checkClass->attribute)==0)

strcpy(set[N].Class,"No");

else

calculate(checkClass->right,N);

}

//Tree height = 2 ==> root->right->right

if(checkClass->cnt==2){

if(checkClass->check > set[N].cost)

strcpy(set[N].Class,"No");

else

strcpy(set[N].Class,"Yes");

}

}