```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

// 시간측정용 변수들
clock_t startTime, endTime;
double nProcessExcuteTime;

// gate
typedef enum { UNDEFINED, PAD, PIN, SOURCE, SINK} VERTEXTYPE;

#define TRUE    1
#define FALSE   0
#define MEMBLKSIZE  128         // 메모리 할당을 위해 쓰이는 블록의 크기
#define BUFNAME     32
#define _CRT_SECURE_NO_WARNINGS
#define INFINITY 30000

struct PATH{
    struct EDGE *path;
    struct PATH *next;
};

struct EDGE{
    int cost;
    int capacity;
    int flow;
    struct VERTEX *prev;
    struct VERTEX *vertex;
    struct EDGE *next;
};

struct VERTEX{
    char name[BUFNAME];
    VERTEXTYPE type;
    int distance;
    struct EDGE *out;
    struct EDGE *predecessor;
    struct VERTEX *next;
};

//초기화
struct VERTEX *headpin = NULL;          // pin vertex의 1번을 가리키는 주소
struct VERTEX *headpad = NULL;      // pad vertex의 1번을 가리키는 주소
struct PATH *headpath = NULL;
struct PATH *matching=NULL;
struct VERTEX *freevert = NULL;         // free vertex 할당을 위한 주소
struct EDGE *freeedge = NULL;           // free edge 할당을 위한 주소
struct PATH *freepath = NULL;           // free edge 할당을 위한 주소

struct VERTEX *source = NULL;           // Source 버텍스
struct VERTEX *sink = NULL;             // Sink 버텍스

struct EDGE *aPath = NULL;


/*************************************************************/

void BuildGraph(char *filename); // graph 생성
struct VERTEX *GetVertex(void);
struct EDGE *GetEdge(void);
struct PATH *GetPath(void);
```

```
inline int Read_a_word(char *str, FILE *fp); //단어 단위로 읽어들임
void vCreateFlowNetwork(void);
void FordFulkerson(void);
int BellmanFord(void);

void vPrintStatus(void);
void vPrintMatching(void);
int iPad, iPin;

int main(void)
{
    char filename[BUFNAME];

    printf("Input file name ? ");
    scanf("%s",filename);
//  sprintf(filename,"input2.txt");

    startTime = clock();

    BuildGraph(filename);
    vCreateFlowNetwork();
    FordFulkerson();
    vPrintMatching();

    endTime = clock();

    nProcessExcuteTime = ( (double)(endTime - startTime) ) / CLOCKS_PER_SEC;
    printf("Excute time: %f second₩n", nProcessExcuteTime);
    return 0;
}

int BellmanFord(void){
    int i=0;
    int s=0;
    int relaxed=1;
    struct VERTEX *tempvert;
    struct EDGE *tempedge;

    // 모든 패드의 거리도 무한대로.
    for (tempvert = headpad; tempvert != NULL; tempvert = tempvert->next) {
        tempvert->distance = INFINITY;
    }
    source -> distance = 0;
    sink ->distance = INFINITY;

    // 패스 초기화
    headpath=NULL;
    // 벨만 포드 알고리즘 ; 모든 버텍스에 사용가능한 엣지를 검사하며 해당 엣지를 추가하였을때 업데이 ↙
    트할 거리정보가 있는지 파악
    while(relaxed){
        relaxed=0;
        for(tempvert = source; tempvert != NULL; tempvert = tempvert->next) {
            for(tempedge = tempvert->out ; tempedge != NULL; tempedge = tempedge->next) {
                if((tempedge->capacity - tempedge->flow) != 0){
                    if((tempedge ->vertex -> distance) > (tempvert ->distance + tempedge -> cost)) {
                        (tempedge ->vertex -> distance) = (tempvert ->distance + tempedge ->cost);
                        tempedge->vertex->predecessor = tempedge;
                        relaxed=1;
                    }
                }
            }
        }
    }
    if(sink->distance == INFINITY)
```

```
        return FALSE;
    else
        return TRUE;
}

void vPrintStatus(void){
    struct VERTEX *tempvert;
    struct EDGE *tempedge;

    for(tempvert=source; tempvert!=NULL; tempvert=tempvert->next){
        printf("%s %d ₩n",tempvert->name,tempvert->distance);
        for(tempedge = tempvert->out; tempedge != NULL ; tempedge = tempedge->next){
            if(tempedge==NULL) break;
            printf(" From %s to %s₩t: cost %d flow %d capacity %d₩n", tempedge->prev->name, tempedge ↙
    ->vertex->name, tempedge->cost, tempedge->flow, tempedge->capacity);
        }
    }
    getchar();
}

void FordFulkerson(void){
    int Cf=INFINITY;
    struct PATH *newpath;
    struct VERTEX *tempvert;
    struct EDGE *tempedge, *tempedge2;

    // 모든 패스가 없을 때까지
    // source 에서 sink로의 경로를 찾는다. in residual nework에서.
    while(BellmanFord()){

        for(tempedge = sink->predecessor; tempedge != NULL ; tempedge = tempedge -> prev ->          ↙
    predecessor){
            if(Cf > (tempedge->capacity - tempedge->flow)) Cf = (tempedge->capacity - tempedge->flow)↙
    ;
        }

        for(tempedge = sink->predecessor; tempedge != NULL ; tempedge = tempedge -> prev ->          ↙
    predecessor){
            tempedge->flow += Cf;
            for(tempedge2=tempedge->vertex->out; tempedge2 != NULL ; tempedge2 = tempedge2 -> next){
                if(strcmp(tempedge->prev->name,tempedge2->vertex->name) == 0)
                    tempedge2->flow = -(tempedge->flow);
            }
        }
    }

    // 맷칭을 구한다 흐름이 1인 것만 모음.
    for(tempvert= headpad; tempvert!=NULL; tempvert= tempvert->next){
        for(tempedge = tempvert->out ; tempedge != NULL ; tempedge = tempedge->next){
            if(tempedge -> flow == 1 && tempedge->vertex->name[0] != 'T' ){
                newpath = GetPath();
                newpath ->path = tempedge;
                newpath ->next = matching;
                matching = newpath;
            }
        }
    }


}

void vPrintMatching(void){
    FILE *fp;
    struct VERTEX * tempvert;
```

```
    struct EDGE *tempedge;
    struct PATH *temppath;
    int totalcost=0;

    fp = fopen("output.txt", "w");
    fprintf(fp,"# of PAD : %d₩n# of PIN : %d₩n₩n", iPad, iPin);

    for(tempvert = headpad ; (tempvert != NULL)&&tempvert->type!=PIN ; tempvert = tempvert->next){
        for(tempedge = tempvert-> out ; tempedge != NULL && (tempedge->vertex->name[0] != 'T') ;
    tempedge = tempedge->next){
            if (tempedge->flow == 1 ) fprintf(fp,"x ");
            else fprintf(fp,"%d ", tempedge->cost);
        }
        fprintf(fp,"₩n");
    }
    fprintf(fp,"₩n");

    fprintf(fp,"PAD₩t-> PIN₩n");

    for(temppath = matching ; temppath != NULL ; temppath = temppath->next) {
        fprintf(fp,"%s₩t-> %s₩n",&temppath->path->prev->name[3], &temppath->path->vertex->name[3]);
        totalcost += temppath->path->cost;
    }
    fprintf(fp,"total cost %d₩n", totalcost);

    fclose(fp);
}
//logic 파일을 읽어들여 그래프 생성
void BuildGraph(char *filename){
    FILE *fp;
    char in,testflag=0;
    int i,  cost;
    char instring[BUFNAME];

    struct VERTEX *newvert, *tempvert,*tempvert2;
    struct EDGE *newedge, *tempedge, *tempedge2;

    //파일 읽기
    if((fp = fopen(filename, "r")) == NULL) {
        printf("ERROR : file can't open!!!₩n");
        exit(0);
    }

    Read_a_word(instring, fp);
    iPad = atoi(instring);
    Read_a_word(instring, fp);
    iPin = atoi(instring);

    for(i = 0; i < iPin; i++){
        newvert = GetVertex();
        newvert->type    =  PIN;
        sprintf(newvert->name,"PIN%d",iPin-i);
        newvert->distance = INFINITY;
        newvert->next = headpin;
        headpin = newvert;
    }

    headpad = headpin;

    // PAD 갯수와 PIN갯수만큼 버텍스를 생성하고 순서대로 이름을 붙여서 정렬한다.
    for(i = 0; i < iPad; i++){
        newvert = GetVertex();
        newvert->type    =  PAD;
        sprintf(newvert->name,"PAD%d",iPad-i);
```

```
        newvert->distance = INFINITY;
        newvert->next = headpad;
        headpad = newvert;
    }

    // PAD와 PIN을 edge로 연결한다.
    tempvert = headpad;
    while(Read_a_word(instring, fp) == TRUE)    {
        if(strcmp(instring,"//") == 0){
            while( ((in=getc(fp)) != EOF) && (in != '₩n') ) ;
            if(in == '₩n') ungetc(in, fp);
        }
        else if(instring[0]>='0' && instring[0]<='9'){
    // 한 줄씩 입력 받음.
            cost = atoi(instring);
            tempedge=tempedge2=NULL;
            // 패드의 갯 수 만큼 코스트가 있을것이므로 패드의 갯수만큼 읽는다.=> 한 줄
            for(tempvert2 = headpin; tempvert2 != NULL; tempvert2 = tempvert2->next)
            {
                if(testflag)
                {
                    Read_a_word(instring, fp);
                    cost=atoi(instring);
                }
                testflag=1;

                newedge = GetEdge();
                newedge->cost = cost;
                newedge->flow = 0;
                newedge->capacity = 1;
                newedge->vertex = tempvert2;
                newedge->prev = tempvert;
                //newedge->next = tempvert->out;
                if(tempvert->out==NULL)
                    tempvert->out = newedge;
                else
                    tempedge->next = newedge;

                tempedge = newedge;

                newedge = GetEdge();
                if(cost != INFINITY)
                    newedge->cost = - cost;
                else
                    newedge->cost = INFINITY;
                newedge->flow = 0 ;
                newedge->capacity = 0;
                newedge->vertex = tempvert;
                newedge->prev = tempvert2;
                newedge->next = tempvert2->out;
                tempvert2->out = newedge;

            }

            tempvert = tempvert->next;
            testflag=0;
        }

    }
    fclose(fp);
} // input

void vCreateFlowNetwork(void){
    struct EDGE *tempedge;
```

```
    struct VERTEX *tempvert;

    source = GetVertex();
    sprintf(source->name,"S");
    source ->type = SOURCE;
    source ->next = headpad;
    sink = GetVertex();
    sprintf(sink->name,"T");
    sink->type=SINK;

    for(tempvert=headpad; tempvert != headpin && tempvert != NULL ; tempvert = tempvert->next){
        tempedge = GetEdge();
        tempedge->cost = 1;
        tempedge->flow = 0;
        tempedge->capacity = 1;
        tempedge->vertex = tempvert;
        tempedge->prev = source;
        tempedge->next = source->out;
        source->out = tempedge;

    }
    // 핀에서 싱크로 가는 그래프를 생성한다.
    // 캐패시티는 1 비용은 1 단방향
    for(tempvert = headpin; tempvert !=NULL ; tempvert = tempvert->next){
        tempedge = GetEdge();
        tempedge->cost =1 ;
        tempedge->flow = 0;
        tempedge->capacity = 1;
        tempedge->vertex = sink;
        tempedge->prev = tempvert;
        tempedge->next = tempvert ->out;
        tempvert ->out = tempedge;
        if(tempvert -> next == NULL) {
            tempvert ->next = sink;
            break;
        }
    }
    // 함수가 끝나면서 source로부터 sink 까지 쭉 연결된 그래프가 된다.
}


struct VERTEX *GetVertex(void)
{
    struct VERTEX *tempvert;
    int i;

    if(freevert == NULL) {
        freevert = (struct VERTEX *)malloc(MEMBLKSIZE*sizeof(struct VERTEX));
        for(i=0;i<MEMBLKSIZE;i++)
            (freevert+i)->next = freevert+i+1;
        (freevert+MEMBLKSIZE-1)->next=NULL;
    }
    tempvert = freevert;
    freevert = freevert ->next;

    tempvert->type = UNDEFINED;
    tempvert->distance = INFINITY;
    tempvert->predecessor = NULL;
    tempvert->out = NULL;
    tempvert->next = NULL;

    return(tempvert);
}
```

```
struct PATH *GetPath(void)
{
    struct PATH *temppath;
    int i;

    if(freepath == NULL) {
        freepath = (struct PATH *)malloc(MEMBLKSIZE*sizeof(struct PATH));
        for(i=0;i<MEMBLKSIZE;i++)
            (freepath+i)->next = freepath+i+1;
        (freepath+MEMBLKSIZE-1)->next=NULL;
    }
    temppath = freepath;
    freepath = freepath->next;

    temppath->next=NULL;
    temppath->path=NULL;

    return(temppath);
}

struct EDGE *GetEdge(void)
{
    struct EDGE *tempedge;
    int i;

    if(freeedge == NULL) {
        freeedge = (struct EDGE *)malloc(MEMBLKSIZE*sizeof(struct EDGE));
        for(i=0;i<MEMBLKSIZE;i++)
            (freeedge+i)->next = freeedge+i+1;
        (freeedge+MEMBLKSIZE-1)->next=NULL;
    }
    tempedge = freeedge;
    freeedge = freeedge->next;

    tempedge->cost = 0;
    tempedge->flow = 0;
    tempedge->capacity = 0;
    tempedge->vertex = NULL;
    tempedge->prev = NULL;
    tempedge->next = NULL;

    return(tempedge);
}

//단어 단위로  읽기
inline int Read_a_word(char *str, FILE *fp)
{
    int        k;
    char       ch;

    while( 1 ) {
        if( (ch=fgetc(fp)) == EOF ) //입력이 없으면
            return( FALSE ); // FALSE를 리턴

        // 이런 것들이 있으면 계속 읽어 나가고
        if( ch=='₩n' || ch==' ' || ch=='₩t' || ch=='(' || ch==')' || ch==',' || ch=='=')
            continue;
        // 이런 게 아니면 빠져 나와라
        else
            break;

    }

    k = 0;
```

```
    str[k] = ch; // 값을 받는다 예를 들어 (3) 같은것의 3을 받는다.
    while( 1 ) {
        if( (ch=fgetc(fp)) == EOF ) //입력이 없으면
            return( FALSE );

        if( ch=='₩n' || ch==' ' || ch=='₩t' || ch=='(' || ch==')' || ch==',' || ch=='=')
            break;
        k ++;

        str[k] = ch;
    }
    k ++;

    str[k] = '₩0'; // 해당사항이 없는 경우는 ₩0를 넣는다.
    return( TRUE );
}
```