

## 과제 #1 Report

김중현/2076088, 곽서진/2076016  
김선영/2071010, 이나현/2076292

### 구현한 C Code

```
/******  
Hashtable Implementation Program  
  
Contributors : 김중현/2076088, 곽서진/2076016, 김선영/2071010, 이나현/2076292  
Date: 03/20/2024  
Assignment : Compiler HW1  
  
*****/  
  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <stdbool.h>  
  
#define FILE_NAME "inputdata1.txt" // 파일명 지정  
#define STsize 1000 // 문자열 테이블 크기  
#define HTsize 100 // 해시 테이블 크기  
#define isLetter(x) (((x) >= 'a' && (x) <= 'z') || ((x) >= 'A' && (x) <= 'Z') || (x) == '_') // 문자  
여부 확인  
#define isDigit(x) ((x) >= '0' && (x) <= '9') // 숫자 여부 확인  
  
typedef struct HTentry* HTpointer;  
typedef struct HTentry {  
    int index; // ST에서 identifier의 인덱스  
    HTpointer next; // 다음 identifier를 가리키는 포인터  
} HTentry;  
  
enum errorTypes { noerror, illsp, illid, illic, overst, overlen }; // 에러 유형 enum으로 정의  
typedef enum errorTypes ERRORtypes;  
  
char separators[] = ".,:;!\\t\\n"; // separator 정의  
  
HTpointer HT[HTsize];  
char ST[STsize];  
  
ERRORtypes error; // 에러 상태
```

```

int nextid = 0; // 현재 identifier 인덱스
int nextfree = 0; // ST에서 다음 available한 인덱스

int hashcode; // identifier의 해시코드
int sameid; // 같은 식별자의 첫 인덱스

bool found; // identifier의 이전 등장 여부

FILE* fp; // 파일을 가리키는 포인터
char input; // 입력 문자

// Input 파일을 여는 함수
void initialize() {
    fp = fopen(FILE_NAME, "r"); // 파일 읽기
    input = fgetc(fp); // 첫 문자 읽기
}

// Separator인지 확인하는 함수
bool isSeparator(char input) {
    int i;
    unsigned long sep_len;

    sep_len = strlen(separators);
    for (i = 0; i < sep_len; i++) {
        if (input == separators[i]) {
            return true; // separator인 경우
        }
    }
    return false; // separator가 아닌 경우
}

// Heading(Index in ST, identifier)을 출력하는 함수
void PrintHeading() {
    printf("-----\n");
    printf(" Index in ST      identifier\n");
    printf("-----\n");
}

// 해시 테이블을 출력하는 함수
void PrintHStable() {
    int i, j;
    HTpointer here; // 현재 노드를 가리킬 포인터

    printf("\n\n\n\n [[ HASH TABLE ]]\n\n");

    for (i = 0; i < HTsize; i++) {
        if (HT[i] != NULL) { // 노드가 존재하면
            printf(" Hash Code %3d : ", i); // 현재 해시 코드 출력
            for (here = HT[i]; here != NULL; here = here->next) { // 연결 리스트의 모든 노드
방문
                j = here->index; // 현재 노드의 identifier 시작 인덱스
                while (ST[j] != '\0' && j < STsize)
                    printf("%c", ST[j++]); // identifier 문자 하나씩 출력
                printf(" ");
            }
        }
    }
}

```

```

    }
    printf("\n");
}
}
printf("\n\n\n < %5d characters are used in the string table > \n", nextfree); // ST에서
사용된 총 문자 수 출력
}

// 발생한 에러에 따라 적절한 에러 메시지를 출력하는 함수
void PrintError(ERRORtypes error) {
    int i;
    switch (error) {
        case noerror: // 에러가 발생하지 않은 경우
            break;
        case overst: // ST 사이즈가 오버플로우인 경우
            printf("...Error...      OVERFLOW ");
            PrintHStable();
            exit(0); // 프로그램 종료
        case illsp: // illegal separator인 경우
            printf("...Error...      %c is illegal separator\n", input);
            break;
        case illid: // input이 숫자로 시작하는 경우
            printf("...Error...      ");
            for(i = nextid; i < nextfree - 1; i++) {
                printf("%c", ST[i]);
            }
            // 출력 정렬을 위해 공백 추가
            // 최대 길이를 20글자로 가정하고, (20 - (nextfree - nextid))만큼 공백 추가
            for (int j = 0; j < 20-(nextfree-nextid); j++) {
                printf(" ");
            }
            printf("start with digit\n");
            nextfree = nextid; // 다음 identifier의 인덱스를 현재 인덱스로 설정하여 identifier
무시
            break;
        case illic: // 허용되지 않은 문자가 나타난 경우
            printf("...Error...      ");
            char illic = '\0'; // 허용되지 않은 문자를 저장할 변수
            for(i = nextid; i < nextfree - 1; i++) {
                printf("%c", ST[i]);
            }
            // 출력 정렬을 위해 공백 추가
            // 최대 길이를 20글자로 가정하고, (20 - (nextfree - nextid))만큼 공백 추가
            for (int j = 0; j < 20-(nextfree-nextid); j++) {
                printf(" ");
            }
            for (i = nextid; i < nextfree - 1; i++) {
                if (!(isDigit(ST[i]) || isLetter(ST[i]))) // 문자가 숫자나 영문자가 아닐 경우
                    illic = ST[i];
            }
            printf("%c is not allowed\n", illic);
            nextfree = nextid;
            break;
        case overlen: // identifier가 12자 이내가 아닌 경우

```

```

    printf("...Error...");
    for (i = nextid; i < nextfree - 1; i++) {
        printf("%c", ST[i]);
    }
    // 출력 정렬을 위해 공백 추가
    // 최대 길이를 20글자로 가정하고, (20 - (nextfree - nextid))만큼 공백 추가
    for (int j = 0; j < 20-(nextfree-nextid); j++) {
        printf(" ");
    }
    printf("too long identifier\n");
    nextfree = nextid;
    break;
}
}

// Separator를 건너뛰는 함수
void SkipSeparators() {
    while (input != EOF && !(isLetter(input) || isDigit(input))) { // input이 파일의 끝에
    도달하지 않았고, 문자나 숫자가 아닐 때
        if (!isSeparator(input)) { // identifier가 아니면
            error = illsp; // illsp(잘못된 공백 문자 에러)로 지정
            PrintError(error);
        }
        input = fgetc(fp); // 다음 문자 읽기
    }
}

// Identifier를 읽는 함수
void ReadID() {
    nextid = nextfree; // input의 시작 index를 nextid에 저장
    if (isDigit(input)) { // input이 숫자로 시작할 때
        error = illid; // illid(잘못된 identifier 에러)로 지정
    }
    while (input != EOF && !isSeparator(input)) { // input이 파일의 끝에 도달하지 않았고
    identifier가 아닐 때까지
        if (nextfree == STsize) { // ST가 꽉 차면
            error = overst; // overst(오버플로우 에러)로 지정
            PrintError(error);
        }
        if(!(isLetter(input) || isDigit(input))) { // input이 영문자나 숫자가 아닌 허용되지 않은
        문자일 경우
            error = illic; // illic(잘못된 문자 에러)로 지정
        }
        ST[nextfree++] = input; // input을 ST에 추가하고 다음 문자로 이동
        input = fgetc(fp); // 다음 문자 읽기
    }
    ST[nextfree++] = '\0'; // 문자열의 끝을 나타내는 null 문자 추가
    PrintError(error);
}

// Hashcode를 계산하는 함수
void ComputeHS(int nid, int nfree) {
    int code, i;
    code = 0;

```

```

    for (i = nid; i < nfree - 1; i++) { // nid부터 nfree-1 까지의 문자에 대해
        code += (int)ST[i]; // 아스키 코드값 합
    }
    hashCode = code % HTsize; // 해시코드값 계산
}

// 해시테이블에서 현재 읽은 identifier가 존재하는지 판단하는 함수
void LookupHS(int nid, int hscode) {
    HTpointer here;
    int i, j;

    found = false; // 초기화
    if (HT[hscode] != NULL) { // 해시테이블의 해당 해시코드에 값이 있으면
        here = HT[hscode]; // 해당 해시코드의 첫 번째 노드 가져오기
        while (here != NULL && found == false) { // 문자가 존재하고 identifier가 발견되지
            않은 경우
                found = true;
                i = here->index; // 현재 노드의 identifier 인덱스
                j = nid; // 읽고있는 identifier의 시작 인덱스
                sameid = i; // 같은 identifier 인덱스 저장

                while (ST[i] != '\0' && ST[j] != '\0' && found == true) { // 문자를 비교하며 identifier
                    일치 여부 판단
                        if (ST[i] != ST[j]) // 문자가 다르면
                            found = false;
                        else { // 다음 문자로 이동
                            i++;
                            j++;
                        }
                    }
                here = here->next; // 연결 리스트의 다음 identifier로 이동
            }
        }
    }
}

// 해시 테이블에 identifier를 추가하는 함수
void ADDHT(int hscode) {
    HTpointer ptr;

    ptr = (HTpointer)malloc(sizeof(ptr)); // 새로운 노드 동적 할당
    ptr->index = nextid; // 현재 identifier의 인덱스 설정
    ptr->next = HT[hscode]; // 새 노드의 다음 노드를 현재 해시코드의 첫 번째 노드로 설정
    HT[hscode] = ptr; // 연결 리스트에 identifier 삽입
}

int main() {
    int i;
    PrintHeading(); // 헤더 출력
    initialize(); // 초기화

    while (input != EOF) { // 파일의 끝에 도달할 때까지
        error = noerror; // 에러 상태 초기화
        SkipSeparators(); // separator 건너뛰기
        ReadID(); // identifier 읽기
    }
}

```

```

    if (error != illid && error != illic) { // 허용되지 않은 문자가 아니며, 숫자로 시작하지
    않는 경우
        if (nextfree == STsize) { // ST가 꽉 찬 경우
            error = overst; // overst(오버플로우)로 에러 지정
            PrintError(error);
        }
        ComputeHS(nextid, nextfree); // 해시코드 계산
        LookupHS(nextid, hashcode); // 해시 테이블에서 identifier 조회
        if (nextfree - nextid > 13) { // identifier의 길이가 12자 이하가 아닌 경우
            error = overlen; // overlen(길이 초과)로 에러 지정
            PrintError(overlen);
        }
        else if (!found) { // 해시 테이블에 동일한 identifier가 존재하지 않는 경우
            printf("%6d      ", nextid); // identifier 인덱스 출력
            printf("      "); // 정렬을 위한 공백
            for (i = nextid; i < nextfree - 1; i++) {
                printf("%c", ST[i]); // identifier 출력
            }
            // 출력 정렬을 위해 공백 추가
            // 최대 길이를 20글자로 가정하고, (20 - (nextfree - nextid))만큼 공백 추가
            for (int j = 0; j < 20-(nextfree-nextid); j++) {
                printf(" ");
            }
            printf("(entered)\n");
            ADDHT(hashcode); // 해시 테이블에 추가
        }
        else { // 해시 테이블에 identifier가 이미 존재하는 경우
            printf("%6d      ", sameid); // 동일한 identifier의 인덱스 출력
            printf("      "); // 정렬을 위한 공백
            for (i = nextid; i < nextfree - 1; i++) {
                printf("%c", ST[i]); // identifier 출력
            }
            // 출력 정렬을 위해 공백 추가
            // 최대 길이를 20글자로 가정하고, (20 - (nextfree - nextid))만큼 공백 추가
            for (int j = 0; j < 20-(nextfree-nextid); j++) {
                printf(" ");
            }
            printf("(already existed)\n");
            nextfree = nextid; // 인덱스 초기화
        }
    }
    SkipSeparators(); // separator 건너뛰기
}
PrintHStable(); // 해시 테이블 출력
printf("김중현/2076088, 곽서진/2076016, 김선영/2071010, 이나현/2076292");
}

```

## 채점용 Test Data Output

### 1. testdata1

#### a. STsize = 1000

 Microsoft Visual Studio 디버그 콘솔

```
-----
Index in ST      identifier
-----
0                wiTHin      (entered)
7                THE         (entered)
11               fLOWer      (entered)
18               gArden      (entered)
25               bUd5        (entered)
30               thattt      (entered)
37               havent      (entered)
44               bl00med      (entered)
52               Endure_each (entered)
64               day         (entered)
68               Amid5t      (entered)
75               hardships   (entered)
85               eAger1y     (entered)
93               awaiting    (entered)
102              their       (entered)
108              M0Ment      (entered)
115              t0          (entered)
118              bl0ss0m     (entered)
```

[[ HASH TABLE ]]

```
Hash Code 7 : Endure_each
Hash Code 12 : bl00med
Hash Code 18 : day
Hash Code 25 : THE
Hash Code 29 : M0Ment
Hash Code 36 : bUd5
Hash Code 40 : their
Hash Code 41 : bl0ss0m
Hash Code 46 : havent
Hash Code 48 : Amid5t
Hash Code 52 : awaiting
Hash Code 54 : eAger1y
Hash Code 60 : fLOWer
Hash Code 64 : t0
Hash Code 65 : thattt
Hash Code 66 : hardships
Hash Code 93 : gArden
Hash Code 95 : wiTHin
```

< 126 characters are used in the string table >

김중현/2076088, 곽서진/2076016, 김선영/2071010, 이나현/2076292

b. STsize = 30

Microsoft Visual Studio 디버그 콘솔

```
-----  
Index in ST      identifier  
-----  
0                wiTHin      (entered)  
7                THE          (entered)  
11               fLOWer       (entered)  
18               gArden       (entered)  
...Error...      OVERFLOW  
  
[[ HASH TABLE ]]  
  
Hash Code 25 : THE  
Hash Code 60 : fLOWer  
Hash Code 93 : gArden  
Hash Code 95 : wiTHin  
  
< 30 characters are used in the string table >
```



## 2. testdata2

Microsoft Visual Studio 디버그 콘솔


```
-----
Index in ST      identifier
-----
      0          wiTHin          (entered)
...Error...      THE&fL_wer      & ls not allowed
      7          gArden          (entered)
     14          bUd5            (entered)
     19          that            (entered)
     24          haventtt        (entered)
     33          bl__med         (entered)
...Error...      ^ is illegal separator
...Error...      Endure#each     # ls not allowed
...Error...      d@y            @ ls not allowed
     41          Amid5t          (entered)
...Error...      ' is illegal separator
...Error...      hardships'     ' ls not allowed
     48          eAgerly         (entered)
     56          awaiting        (entered)
     65          their           (entered)
     71          M_Ment          (entered)
     78          t0              (entered)
     81          bl_ss_m         (entered)

[[ HASH TABLE ]]

Hash Code   6 : bl__med
Hash Code  33 : that
Hash Code  35 : bl_ss_m
Hash Code  36 : bUd5
Hash Code  40 : their
Hash Code  48 : Amid5t
Hash Code  52 : awaiting
Hash Code  54 : eAgerly
Hash Code  64 : t0
Hash Code  76 : M_Ment
Hash Code  78 : haventtt
Hash Code  93 : gArden
Hash Code  95 : wiTHin

< 89 characters are used in the string table >
김중현/2076088, 곽서진/2076016, 김선영/2071010, 이나현/2076292
```

### 3. testdata3

 Microsoft Visual Studio 디버그 콘솔

```
-----
Index in ST      identifier
-----
      0          ThEy          (entered)
      5          w1thstAnd      (entered)
     15          be1Ng_SHAken    (entered)
     28          AND            (entered)
     32          leaning         (entered)
     40          NEVER          (entered)
     46          breaking        (entered)
     55          in             (entered)
     55          in             (already existed)
     58          Overc0ming      (entered)
     69          ANd            (entered)
     73          they           (entered)
...Error...      f1na1ly_b100m    too long identifier
     78          in2            (entered)
     82          fLOWers_        (entered)
```


[[ HASH TABLE ]]

```
Hash Code 11 : AND
Hash Code 15 : in
Hash Code 17 : Overc0ming
Hash Code 34 : leaning
Hash Code 35 : breaking
Hash Code 42 : they
Hash Code 43 : ANd
Hash Code 62 : be1Ng_SHAken
Hash Code 65 : in2
Hash Code 70 : fLOWers_
Hash Code 78 : ThEy
Hash Code 84 : NEVER
Hash Code 94 : w1thstAnd
```

< 91 characters are used in the string table >

김중현/2076088, 광서진/2076016, 김선영/2071010, 이나현/2076292

#### 4. testdata4

 Microsoft Visual Studio 디버그 콘솔

```
-----
Index in ST      identifier
-----
      0          Th2s2          (entered)
      6          flowers        (entered)
     14          hav1ng         (entered)
     21          bLo0mEd        (entered)
...Error...      5o             start with digit
     29          bEAUtlfU1ly    (entered)
     41          through        (entered)
     49          adversity      (entered)
...Error...      52remind       start with digit
     59          me             (entered)
...Error...      0f             start with digit
     62          a              (entered)
     64          girl           (entered)
     69          i              (entered)
     71          know           (entered)
     76          l              (entered)
     78          call           (entered)
     83          ThEm           (entered)
     88          Ewha           (entered)
```

[[ HASH TABLE ]]

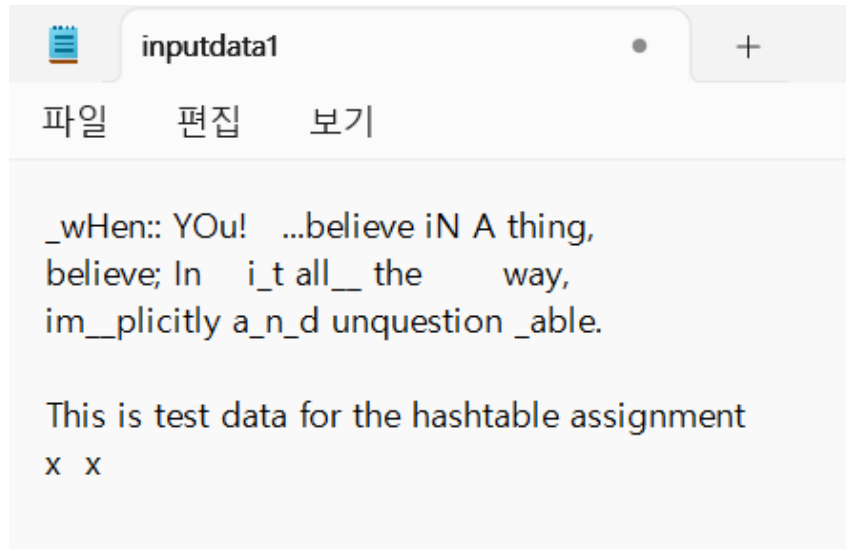
```
Hash Code   3 : Th2s2
Hash Code   5 : i
Hash Code  10 : me
Hash Code  11 : bLo0mEd
Hash Code  12 : bEAUtlfU1ly
Hash Code  30 : girl
Hash Code  47 : know
Hash Code  66 : ThEm
Hash Code  69 : through
Hash Code  70 : flowers
Hash Code  73 : l
Hash Code  81 : hav1ng
Hash Code  87 : adversity
Hash Code  89 : Ewha
Hash Code  94 : call
Hash Code  97 : a
```

< 93 characters are used in the string table >

김중현/2076088, 곽서진/2076016, 김선영/2071010, 이나현/2076292

직접 작성한 에러 없는 입력데이터(**inputdata1, inputdata2, inputdata3**)

1. inputdata1  
**input**



**output**(다음 장)

```
-----
Index in ST      identifier
-----
0                _when      (entered)
6                YOU        (entered)
10               believe    (entered)
18               iN         (entered)
21               A          (entered)
23               thing      (entered)
10               believe    (already existed)
29               In         (entered)
32               i_t        (entered)
36               all__      (entered)
42               the        (entered)
46               way        (entered)
50               im__plicitly (entered)
63               a_n_d      (entered)
69               unquestion (entered)
80               _able      (entered)
86               This       (entered)
91               is         (entered)
94               test       (entered)
99               data       (entered)
104              for        (entered)
42               the        (already existed)
108              hashtable  (entered)
118              assignment (entered)
129              x          (entered)
129              x          (already existed)

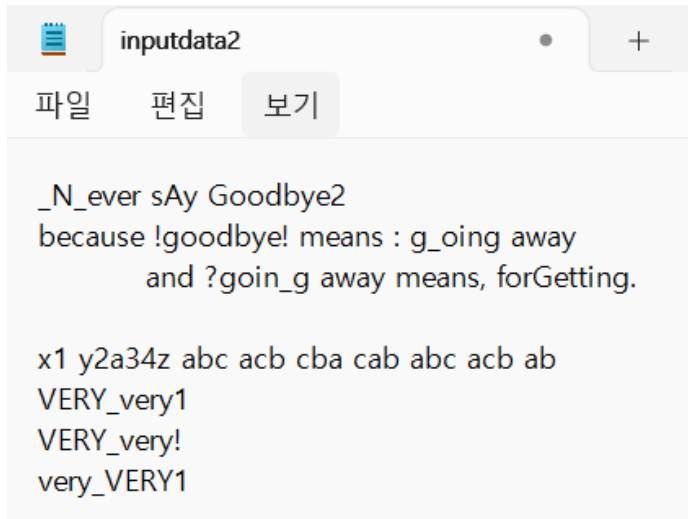
[[ HASH TABLE ]]

Hash Code 3 : all__
Hash Code 8 : This
Hash Code 10 : data
Hash Code 15 : unquestion
Hash Code 16 : i_t
Hash Code 20 : x is
Hash Code 21 : the
Hash Code 27 : for
Hash Code 32 : believe
Hash Code 37 : way
Hash Code 38 : thing
Hash Code 40 : hashtable
Hash Code 48 : test
Hash Code 65 : A
Hash Code 78 : im__plicitly
Hash Code 81 : assignment
Hash Code 83 : In iN
Hash Code 85 : YOU
Hash Code 97 : a_n_d _when
Hash Code 99 : _able

< 131 characters are used in the string table >
김중현/2076088, 박서진/2076016, 김선영/2071010, 이나현/2076292
```

## 2. inputdata2

### input



```
_N_ever sAy Goodbye2  
because !goodbye! means : g_oing away  
and ?goin_g away means, forGetting.  
  
x1 y2a34z abc acb cba cab abc acb ab  
VERY_very1  
VERY_very!  
very_VERY1
```

### output(다음 장)

```

-----
Index in ST      identifier
-----
0                _N_ever      (entered)
8                sAy          (entered)
12               Goodbye2     (entered)
21               because     (entered)
29               goodbye     (entered)
37               means       (entered)
43               g_oing      (entered)
50               away        (entered)
55               and         (entered)
59               goin_g      (entered)
50               away        (already existed)
37               means       (already existed)
66               forGetting  (entered)
77               x1          (entered)
80               y2a34z      (entered)
87               abc         (entered)
91               acb         (entered)
95               cba         (entered)
99               cab         (entered)
87               abc         (already existed)
91               acb         (already existed)
103              ab          (entered)
106              VERY_very1  (entered)
117              VERY_very   (entered)
127              very_VERY1  (entered)

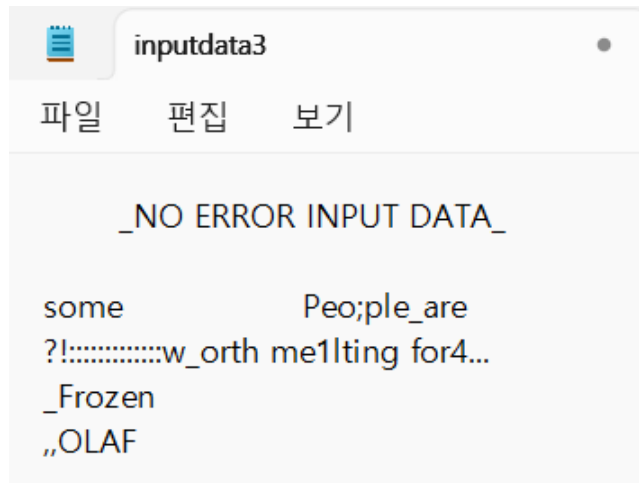
[[ HASH TABLE ]]

Hash Code 1 : sAy
Hash Code 2 : _N_ever
Hash Code 7 : and
Hash Code 24 : very_VERY1    VERY_very1
Hash Code 27 : goin_g      g_oing
Hash Code 28 : because
Hash Code 32 : means
Hash Code 34 : away
Hash Code 45 : goodbye
Hash Code 49 : forGetting
Hash Code 63 : Goodbye2
Hash Code 69 : x1
Hash Code 75 : VERY_very
Hash Code 93 : y2a34z
Hash Code 94 : cab      cba      acb      abc
Hash Code 95 : ab

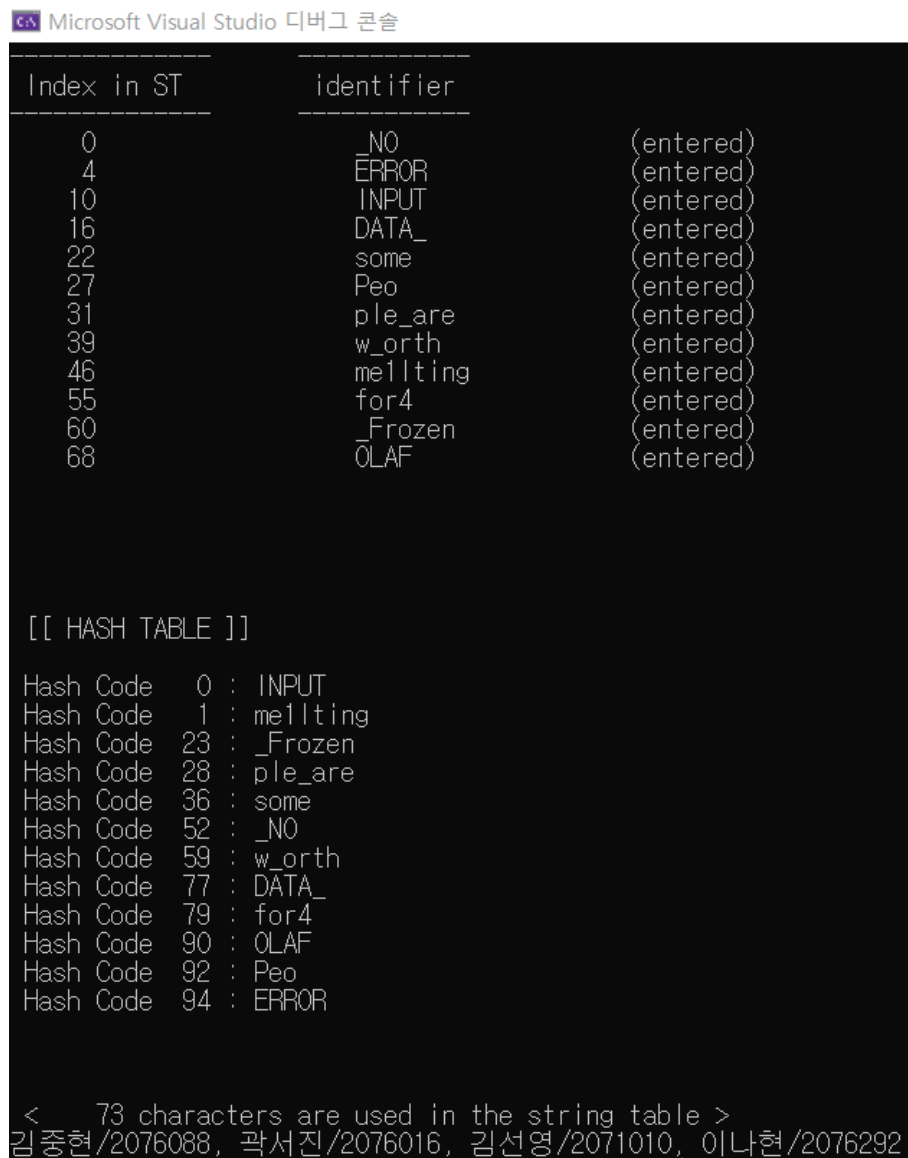
< 138 characters are used in the string table >
김중현/2076088, 광서진/2076016, 김선영/2071010, 이나현/2076292

```

### 3. inputdata3 input



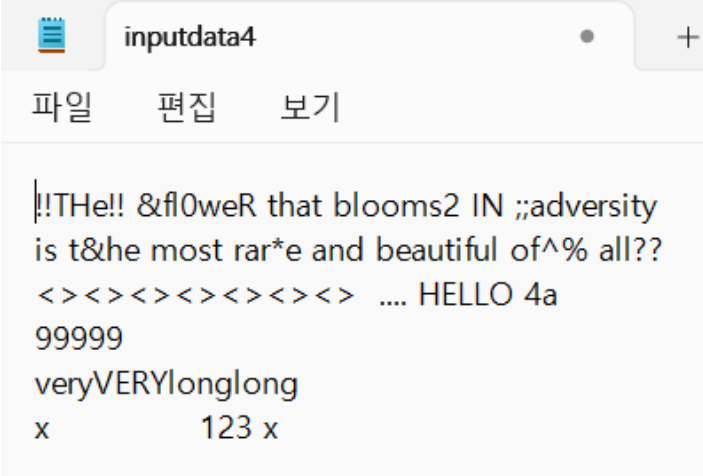
### output





직접 작성한 에러 있는 입력데이터(inputdata4, inputdata5, inputdata6)

1. inputdata4  
input



```
!!The!! &f!0weR that blooms2 IN ;;adversity
is t&he most rar*e and beautiful of^% all??
<><><><><><><> .... HELLO 4a
99999
veryVERYlonglong
x          123 x
```

output(다음 장)

```

-----
Index in ST      identifier
-----
0               The (entered)
...Error...     & is illegal separator
4               f10weR (entered)
11              that (entered)
16              blooms2 (entered)
24              IN (entered)
27              adversity (entered)
37              is (entered)
...Error...     t&he & ls not allowed
40              most (entered)
...Error...     rar*e * ls not allowed
45              and (entered)
49              beautiful (entered)
...Error...     of^% % ls not allowed
59              all (entered)
...Error...     < is illegal separator
...Error...     > is illegal separator
...Error...     < is illegal separator
...Error...     > is illegal separator
...Error...     < is illegal separator
...Error...     > is illegal separator
...Error...     < is illegal separator
...Error...     > is illegal separator
...Error...     < is illegal separator
...Error...     > is illegal separator
...Error...     < is illegal separator
...Error...     > is illegal separator
...Error...     < is illegal separator
...Error...     > is illegal separator
63              HELLO (entered)
...Error...     4a start with digit
...Error...     99999 start with digit
...Error...     veryVERYlonglong too long identifier
69              x (entered)
...Error...     123 start with digit
69              x (already existed)

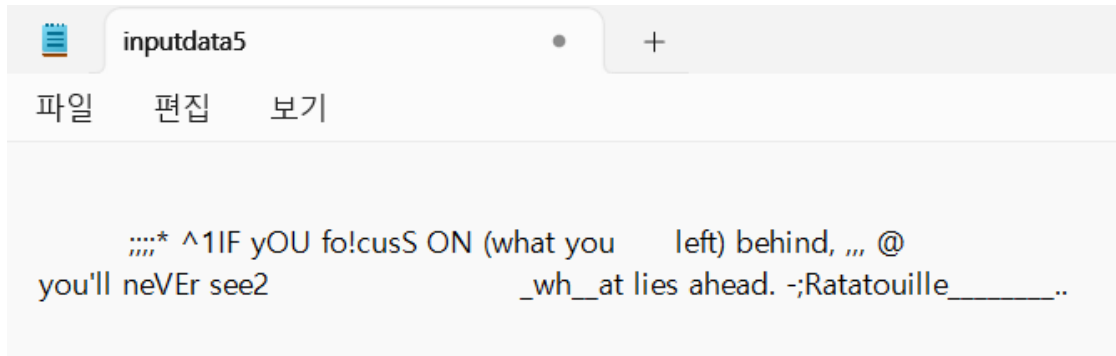
[[ HASH TABLE ]]

Hash Code 2 : blooms2
Hash Code 7 : and
Hash Code 13 : all
Hash Code 20 : x is
Hash Code 33 : that
Hash Code 51 : most IN
Hash Code 57 : The
Hash Code 60 : f10weR
Hash Code 61 : beautiful
Hash Code 72 : HELLO
Hash Code 87 : adversity

< 71 characters are used in the string table >
김중현/2076088, 광서진/2076016, 김선영/2071010, 이나현/2076292,

```

2. inputdata5  
input



The screenshot shows a code editor window with the title 'inputdata5'. Below the title bar is a menu bar with three items: '파일' (File), '편집' (Edit), and '보기' (View). The main editing area contains the following text:

```
;;;;* ^1IF yOU fo!cusS ON (what you left) behind, ,,, @  
you'll neVEr see2 _wh__at lies ahead. -;Ratatouille_____..
```

output(다음 장)

```

-----
Index in ST      identifier
-----
...Error...      * is illegal separator
...Error...      ^ is illegal separator
...Error...      1IF start with digit
0                yOU      (entered)
4                fo       (entered)
7                cusS     (entered)
12               ON       (entered)
...Error...      ( is illegal separator
15               what     (entered)
20               you      (entered)
...Error...      left)      ) is not allowed
24               behind   (entered)
...Error...      @ is illegal separator
...Error...      you'll    ' is not allowed
31               neVEr    (entered)
37               see2     (entered)
42               _wh__at  (entered)
50               lies     (entered)
55               ahead    (entered)
...Error...      - is illegal separator
...Error...      Ratatouille_____too long identifier

```

# [[ HASH TABLE ]]

```

Hash Code 13 : fo
Hash Code 14 : cusS
Hash Code 18 : behind
Hash Code 21 : _wh__at
Hash Code 29 : lies
Hash Code 36 : what
Hash Code 49 : you
Hash Code 57 : ON
Hash Code 67 : see2
Hash Code 80 : neVEr
Hash Code 85 : yOU
Hash Code 99 : ahead

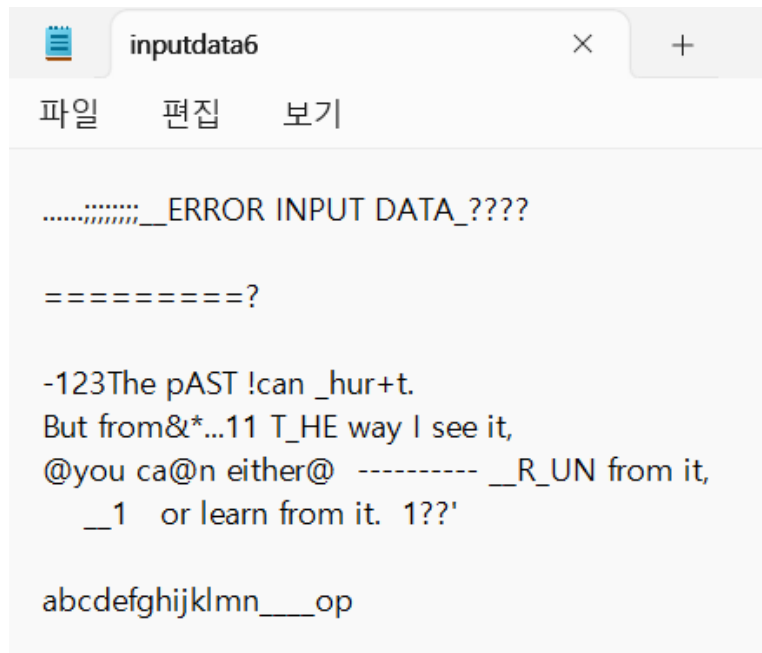
```

< 61 characters are used in the string table >

김중현/2076088, 광서진/2076016, 김선영/2071010, 이나현/2076292

### 3. inputdata6

input



output(다음 장)

```

-----
Index in ST      identifier
-----
0                __ERROR      (entered)
8                INPUT        (entered)
14              DATA_        (entered)
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      = is illegal separator
...Error...      123The      start with digit
20              pAST          (entered)
25              can           (entered)
...Error...      _hur+t       + is not allowed
29              But           (entered)
...Error...      from&*       * is not allowed
...Error...      11           start with digit
33              T_HE          (entered)
38              way           (entered)
42              |             (entered)
44              see           (entered)
48              it            (entered)
...Error...      @ is illegal separator
51              you           (entered)
...Error...      ca@n         @ is not allowed
...Error...      either@      @ is not allowed
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
...Error...      - is illegal separator
55              __R_UN        (entered)
62              from          (entered)
48              it            (already existed)
67              __1           (entered)
71              or            (entered)
74              learn         (entered)
62              from          (already existed)
48              it            (already existed)
...Error...      1            start with digit
...Error...      ' is illegal separator
...Error...      abcdefghijklmn____optoo long identifier

```

```
[[ HASH TABLE ]]
```

```
Hash Code  0 : INPUT
Hash Code  6 : can
Hash Code 17 : see
Hash Code 20 : T_HE
Hash Code 21 : it
Hash Code 25 : or
Hash Code 30 : learn    __R_UN
Hash Code 36 : from
Hash Code 37 : way
Hash Code 39 : __1
Hash Code 44 : pAST
Hash Code 49 : you
Hash Code 73 : I
Hash Code 77 : DATA_
Hash Code 84 : __ERROR
Hash Code 99 : But
```

```
< 80 characters are used in the string table >
```

```
김중현/2076088, 곽서진/2076016, 김선영/2071010, 이나현/2076292
```

## 과제 정의에 대한 추가 설명

1. **identifier**에서 허용되지 않은 문자가 문자열의 가장 앞에 존재할 경우 **SkipSeparators()** 함수를 통해 **illegal separator**로 처리하고 이후 오는 문자열을 검사한다.

ex) 'hardships' 문자에 대해 ' 문자를 **illegal separator**로, hardships'는 ' is not allowed 에러로 처리한다.

2. 출력 정렬을 맞추기 위해 공백을 추가한다. 이 때, 입력으로 들어오는 **input**에서 한 **identifier**의 최대 길이를 20글자로 가정하고, **(20-(nextfree-nextid))**만큼 공백을 추가한다.

```
for (int j = 0; j < 20-(nextfree-nextid); j++) {
    printf(" ");
}
```