컴파일러 구성 과제#1

1605020 박소현

1642041 이경연

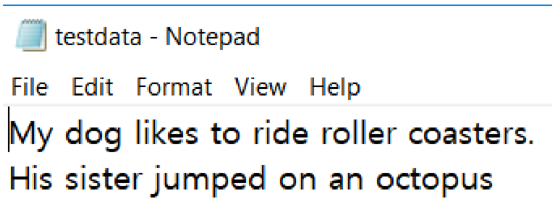
목차

1.Code

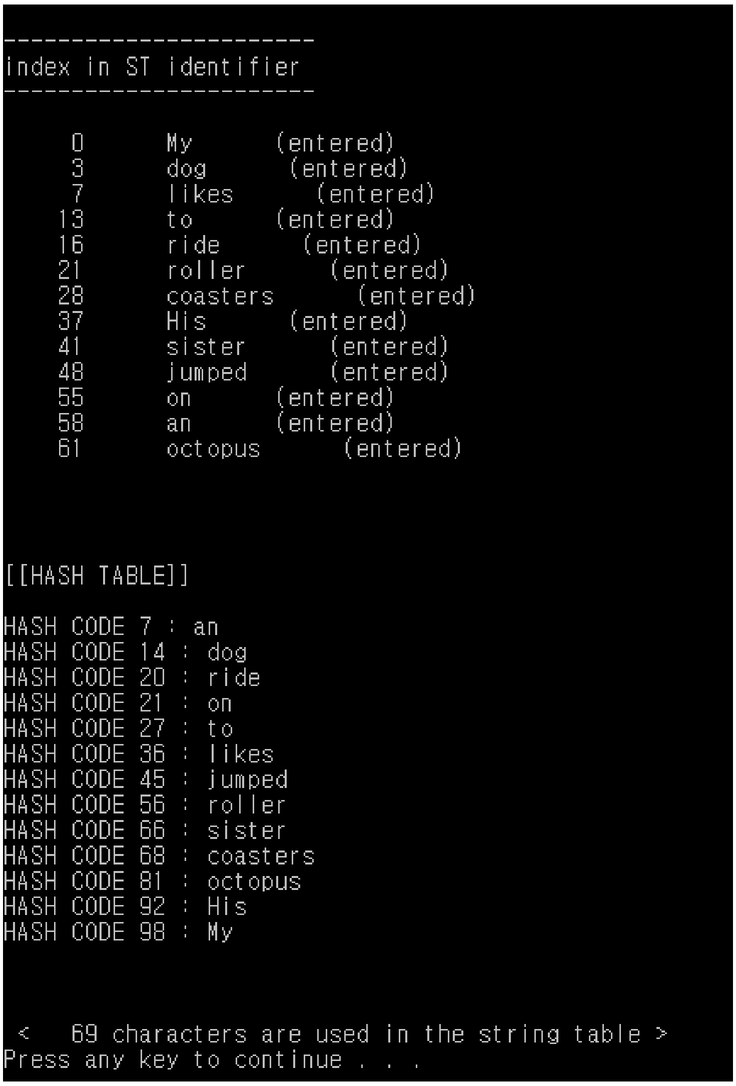
```
-   #include <stdio.h>
-   #include <stdlib.h>
-   #include <string.h>
-
-   #define FILE_NAME "testdata1.txt"
-
-   #define STsize 30   //size of string table
-   #define HTsize 100   //size of hash table
-
-   #define FALSE 0
-   #define TRUE 1
-
-   //letter, digit, 그리고 seperator에 대해 특정 문자가 해당 변수에 속하는지 체크할 때
    사용
-   #define isLetter(ch) (((ch) >= 'a' && (ch) <= 'z') || ((ch) >= 'A' && (ch) <= 'Z') ||
    (ch) == '_')
-   #define isDigit(x) (((x)<='9' && (x)>='0'))
-   #define isSeperator(s) ((s) == ' ' | (s) == '₩n' | (s) == '₩t' | (s) == ';' | (s) ==
    '.' | (s) == ',' | (s) == '?' | (s) == '!')
-
-   typedef struct HTentry *HTpointer;
-   typedef struct HTentry {
-    int index;   //index of identifier in ST
-    HTpointer next;   //pointer to next identifier
-   }HTentry;
-
-   //나올 수 있는 여러 종류의 에러타입들을 미리 지정해둔다.
-   enum errorTypes { noerror, illsp, illid, overst, toolong };
-   typedef enum errorTypes ERRORtypes;
-   HTpointer HT[HTsize];
-   char ST[STsize];
-
-   int nextid = 0;
-   int nextfree = 0;
-   int hashcode = 0;
-   int sameid = 0;
-   int found;
-
-   ERRORtypes err;
-
-   FILE *fp;
-   char input;
-
-   //file을 읽기 모드로 열어준 뒤, 해당 파일에서 문자 한 개를 읽어오는 함수
-   void initialize() {
-    fp = fopen(FILE_NAME, "r");
-    input = fgetc(fp);
-   }
-
-   void PrintHeading() {
-    printf("₩n₩n");
-    printf("-----------------------₩n");
-    printf("index in ST identifier₩n");
-    printf("-----------------------₩n");
-    printf("₩n");
-   }
-
-   //Hash Table에 저장되어진 값들을 출력해주는 함수
-   void PrintHStable() {
-    int i, j;
-    HTpointer here;
-    printf("₩n₩n₩n₩n[[HASH TABLE]]₩n₩n");
-
-    //Hash Table의 크기만큼 반복
-    for (i = 0; i < HTsize; i++)
-            if (HT[i] != NULL) {
-                    printf("HASH CODE %d : ", i);
-                    //Hash Table에서 here가 NULL이 될 때까지 here->next를 통해 다음으로
    계속 반복
-                    for (here = HT[i]; here != NULL; here = here->next) {
-                            j = here->index;
-                            //String table에서 빈 칸이 나올 때까지 저장되어있는
    문자값들을 출력함.
-                            while (j < STsize && ST[j] != '₩0') {
-                                    printf("%c", ST[j++]);
-                            }
-                            printf("   ");
-                    }
-                    printf("₩n");
-            }
-    printf("₩n₩n₩n <%5d characters are used in the string table > ₩n", nextfree);
-   }
-
-   //err에 저장된 값에 따라 다른 에러문이 출력되도록 해주는 함수
-   void PrintError(ERRORtypes err) {
-   //String table의 크기보다 더 많은 값들이 저장된 경우
-   if (err == overst) {
-           printf("...Error... OVERFLOW");
-           PrintHStable();
-   }
-   else if (err == illsp)   //지정해준 seperator가 아닌 다른 문자가 입력될 경우
-           printf("...Error... %c is illegal seperator ₩n", input);
-   else if (err == toolong)   //변수의 길이가 너무 긴 경우
-           printf("...Error... ");
-   //변수 이름이 숫자로 시작되는 경우
-   else if (err == illid) {
-           printf("...Error... ");
-           while (input != EOF && (isLetter(input) || isDigit(input)))
-           {
-                   printf("%c", input);
-                   input = fgetc(fp);
-           }
-           printf("₩tStart with digit₩n");
-   }
-   }
-
-   //seperator 값이 입력되는 경우 다음 문자를 읽어들이는 함수
-   void SkipSeperators() {
-   while (input != EOF && !(isLetter(input) || isDigit(input)))
-   {
-           //지정된 seperator가 아닐 경우에는 에러 출력 함수를 호출
-           if (!isSeperator(input)) {
-                   err = illsp;
-                   PrintError(err);
-           }
-           input = fgetc(fp);
-   }
-   }
-
-   //file로부터 문자들을 한 개씩 읽어와 string table에 저장해주는 함수
-   void ReadID() {
-   nextid = nextfree;
-   //읽어들인 input값이 숫자로 시작되는 경우 에러 출력 함수를 호출
-   if (isDigit(input)) {
-           err = illid;
-           PrintError(err);
-   }
-   else {
-           //input이 문자 또는 숫자일 경우 string table에 해당 값을 저장해준 뒤
    nextfree를 1 증가시키고 file로부터 새로 읽어온다.(이를 반복)
-           while ((isLetter(input) || isDigit(input))) {
-                   //nextfree가 STsize값과 똑같을 경우 string table이 가득 찼다는
    얘기이므로 overflow 에러를 출력시키도록 에러 출력 함수를 호출한다.
-                   if (nextfree == STsize) {
-                           err = overst;
-                           PrintError(err);
-                   }
-                   ST[nextfree] = input;
-                   nextfree++;
-                   input = fgetc(fp);
-           }
-   }
-   }
-
-   //hash code를 계산해주는 함수
-   void ComputeHS(int nid, int nfree) {
-   int code, i;
-   code = 0;
-   //string table에 저장되어있는 변수 길이만큼 for문을 반복하며 값을 더해준다.(이 때,
    저장되어있는 값은 문자이므로 정수값으로 바꿔줘야 한다.)
-   for (i = nid; i < nfree - 1; i++)
-           code += (int)ST[i];
-   //hash code값이 hash table의 크기를 벗어나면 안되므로 HTsize를 나눈 나머지를 hash
    code로 넣는다.
-   hashcode = code % HTsize;
-   }
-
-   //hash table에 해당 hash code값이 있는지를 확인해주는 함수
-   void LookupHS(int nid, int hscode) {
-   HTpointer here;
-   int i, j;
-   found = FALSE; //found를 먼저 FALSE로 초기화시켜준다.
-
-   //hash table에서 hash code 값이 있을 경우
-   if (HT[hscode] != NULL) {
-           //hash table에 here 포인터 배치
-           here = HT[hscode];
-           while (here != NULL && found == FALSE) {
-                   found = TRUE;
-                   i = here->index;
-                   j = nid;
-                       sameid = i;
-
-                       //다음 identifier까지 index 조정
-                       while (ST[i] != '₩0' && ST[j] != '₩0' && found == TRUE) {
-                               if (ST[i] != ST[j])
-                                       found = FALSE;
-                               else {
-                                       i++;
-                                       j++;
-                               }
-                       }
-                       //hash table 다음 칸으로 조정
-                       here = here->next;
-               }
-       }
-   }
-
-   //계산된 hash code값을 hash table에 저장해주는 함수
-   void ADDHT(int hscode) {
-   HTpointer ptr;
-   ptr = (HTpointer)malloc(sizeof(ptr));   //HTpointer의 크기만큼 메모리 공간을
    할당해준다.
-   ptr->index = nextid;
-   ptr->next = HT[hscode];
-   HT[hscode] = ptr;
-   }
-
-
-   int main() {
-   int i;
-   PrintHeading();
-   initialize();
-   //파일의 끝에 도달하기 전까지 while문 반복
-   while (input != EOF)
-   {
-           err = noerror;
-           SkipSeperators();
-           ReadID();
-
-           //잘못된 변수가 아닐 경우
-           if (err != illid) {
-                   if (nextfree == STsize) {
-                           err = overst;
-                               PrintError(err);
-                       }
-                       ST[nextfree++] = '₩0';
-                       ComputeHS(nextid, nextfree); //읽어온 값들에 대해 hash code를 계산
-                       LookupHS(nextid, hashcode);   //계산된 hash code값이 hash table에
    이미 있는지 확인
-
-                       //hash table에 해당 hash code값이 없을 경우
-                       if (!found&&ST[nextid] != NULL) {
-                               //변수의 길이가 너무 길지 않을 경우
-                               if ((nextfree - nextid - 1) <= 10) {
-                                       printf("%6d      ", nextid);
-                                       for (i = nextid; i < nextfree - 1; i++) //string
    table에 저장되어있는 변수를 출력해준다.
-                                               printf("%c", ST[i]);
-                                       printf("      (entered)₩n");
-                                       ADDHT(hashcode); //해당 hash code를 hash table에
    저장해준다.
-                               }
-                               //변수의 길이가 너무 길 경우
-                               else  if ((nextfree - nextid - 1) > 10) {
-                                       err = toolong;
-                                       PrintError(err); //변수 길이가 긴 것에 대한 에러를
    출력해준다.
-                                       for (i = nextid; i < nextfree - 1; i++)
-                                               printf("%c", ST[i]);
-                                       nextfree = nextid; //string table에 변수가 저장되지
    않도록 nextid값을 다시 nextfree에 넣어준다.
-                                       printf("₩t too long identifier₩n");
-                               }
-                       }
-                       //hash table에 해당 hash code값이 있는 경우
-                       else if (found) {
-                               printf("%6d      ", sameid);
-                               for (i = nextid; i < nextfree - 1; i++)
-                                       printf("%c", ST[i]);
-                               printf("      (already existed)₩n");
-                               nextfree = nextid; //string table에 변수가 저장되지 않도록
    nextid값을 다시 nextfree에 넣어준다.
-                       }
-               }
-       }
-       PrintHStable(); //모든 것이 완료된 후 hash table을 출력해준다.
-       }
```

2.No error in the input data file

2.1 Input data_1

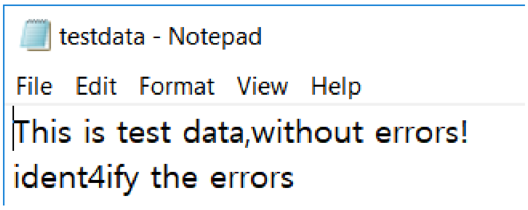


My dog likes to ride roller coasters.
His sister jumped on an octopus

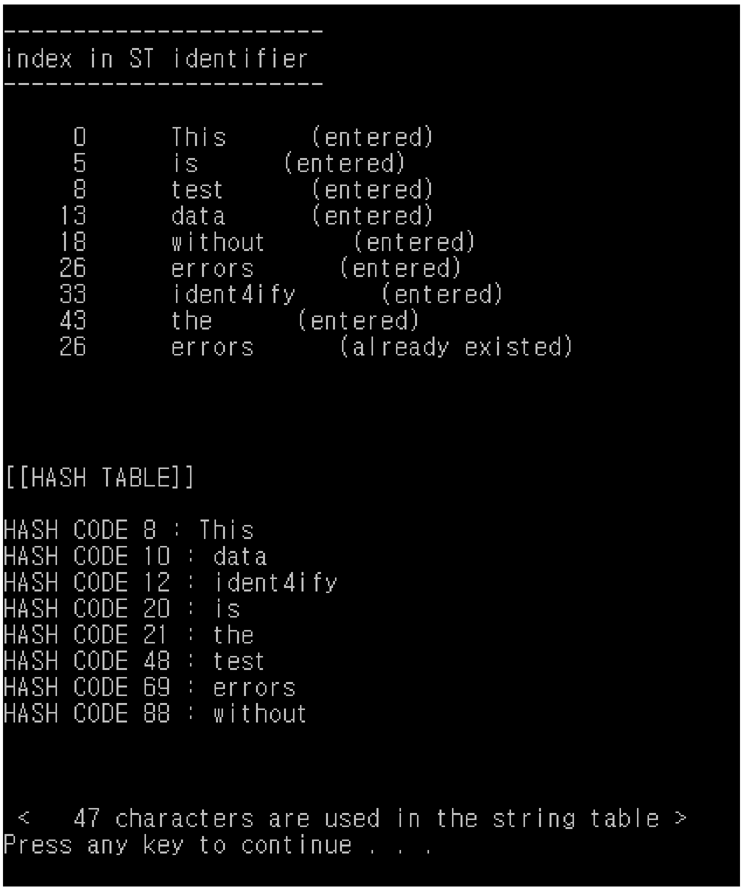


```
-----------------------
index in ST identifier
-----------------------

     0      My      (entered)
     3      dog      (entered)
     7      likes      (entered)
    13      to      (entered)
    16      ride      (entered)
    21      roller      (entered)
    28      coasters      (entered)
    37      His      (entered)
    41      sister      (entered)
    48      jumped      (entered)
    55      on      (entered)
    58      an      (entered)
    61      octopus      (entered)




[[HASH TABLE]]

HASH CODE 7 : an
HASH CODE 14 : dog
HASH CODE 20 : ride
HASH CODE 21 : on
HASH CODE 27 : to
HASH CODE 36 : likes
HASH CODE 45 : jumped
HASH CODE 56 : roller
HASH CODE 66 : sister
HASH CODE 68 : coasters
HASH CODE 81 : octopus
HASH CODE 92 : His
HASH CODE 98 : My



 <   69 characters are used in the string table >
Press any key to continue . . .
```

2.2 Input data_2

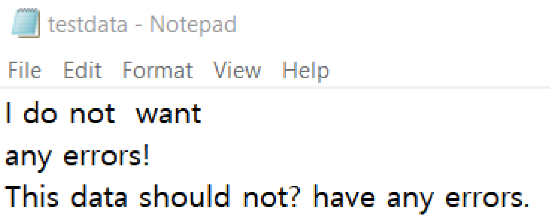


```
This is test data,without errors!
ident4ify the errors
```

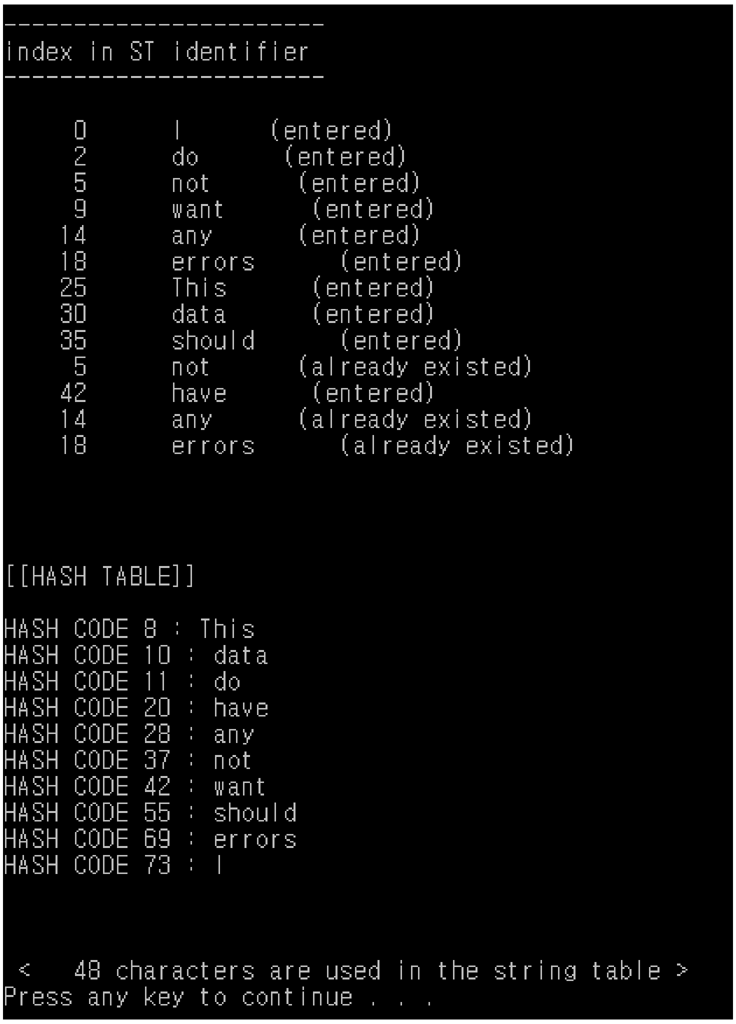


```
-----------------------
index in ST identifier
-----------------------

     0      This      (entered)
     5      is      (entered)
     8      test      (entered)
    13      data      (entered)
    18      without      (entered)
    26      errors      (entered)
    33      ident4ify      (entered)
    43      the      (entered)
    26      errors      (already existed)



[[HASH TABLE]]

HASH CODE 8 : This
HASH CODE 10 : data
HASH CODE 12 : ident4ify
HASH CODE 20 : is
HASH CODE 21 : the
HASH CODE 48 : test
HASH CODE 69 : errors
HASH CODE 88 : without



 <   47 characters are used in the string table >
Press any key to continue . . .
```

2.3 Input data_3



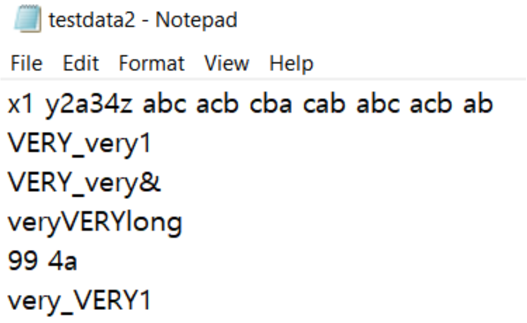
```
I do not  want
any errors!
This data should not? have any errors.
```



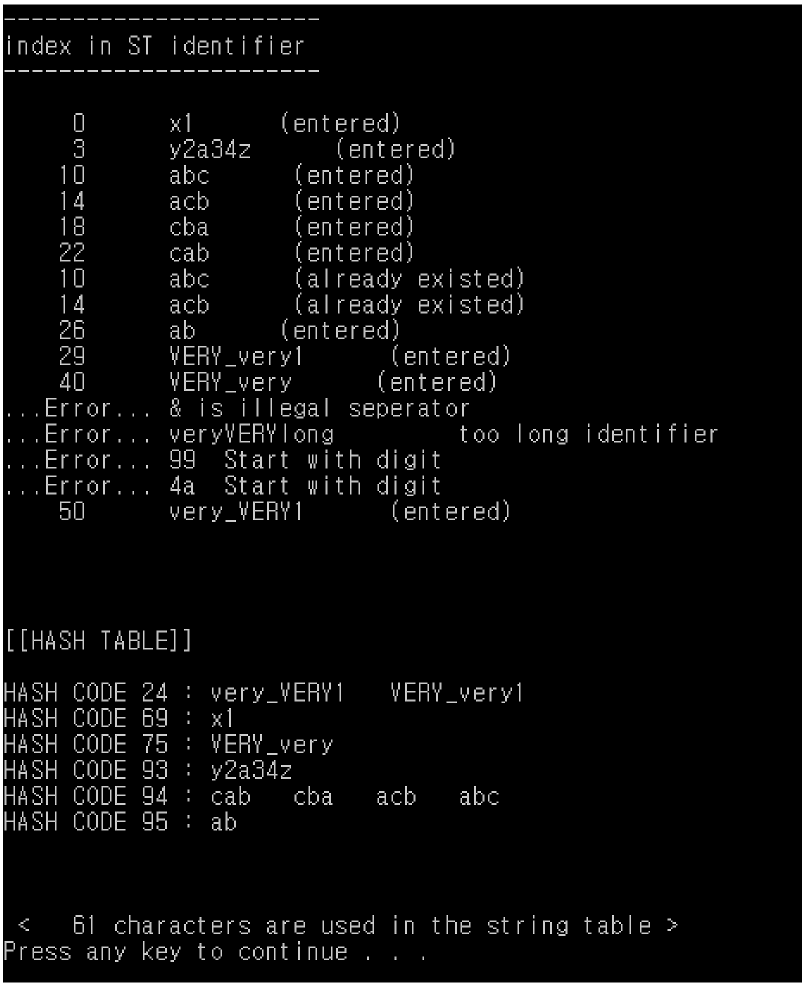
```
-----------------------
index in ST identifier
-----------------------

     0      I      (entered)
     2      do      (entered)
     5      not      (entered)
     9      want      (entered)
    14      any      (entered)
    18      errors      (entered)
    25      This      (entered)
    30      data      (entered)
    35      should      (entered)
     5      not      (already existed)
    42      have      (entered)
    14      any      (already existed)
    18      errors      (already existed)



[[HASH TABLE]]

HASH CODE 8 : This
HASH CODE 10 : data
HASH CODE 11 : do
HASH CODE 20 : have
HASH CODE 28 : any
HASH CODE 37 : not
HASH CODE 42 : want
HASH CODE 55 : should
HASH CODE 69 : errors
HASH CODE 73 : I



 <   48 characters are used in the string table >
Press any key to continue . . .
```

3.With error in the input data file

3.1 Input data_1

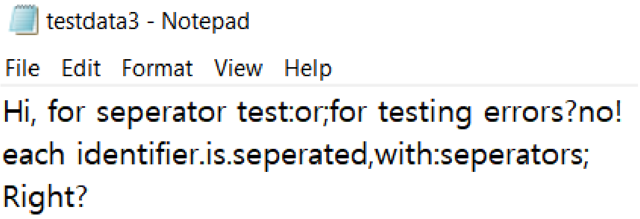


```
x1 y2a34z abc acb cba cab abc acb ab
VERY_very1
VERY_very&
veryVERYlong
99 4a
very_VERY1
```

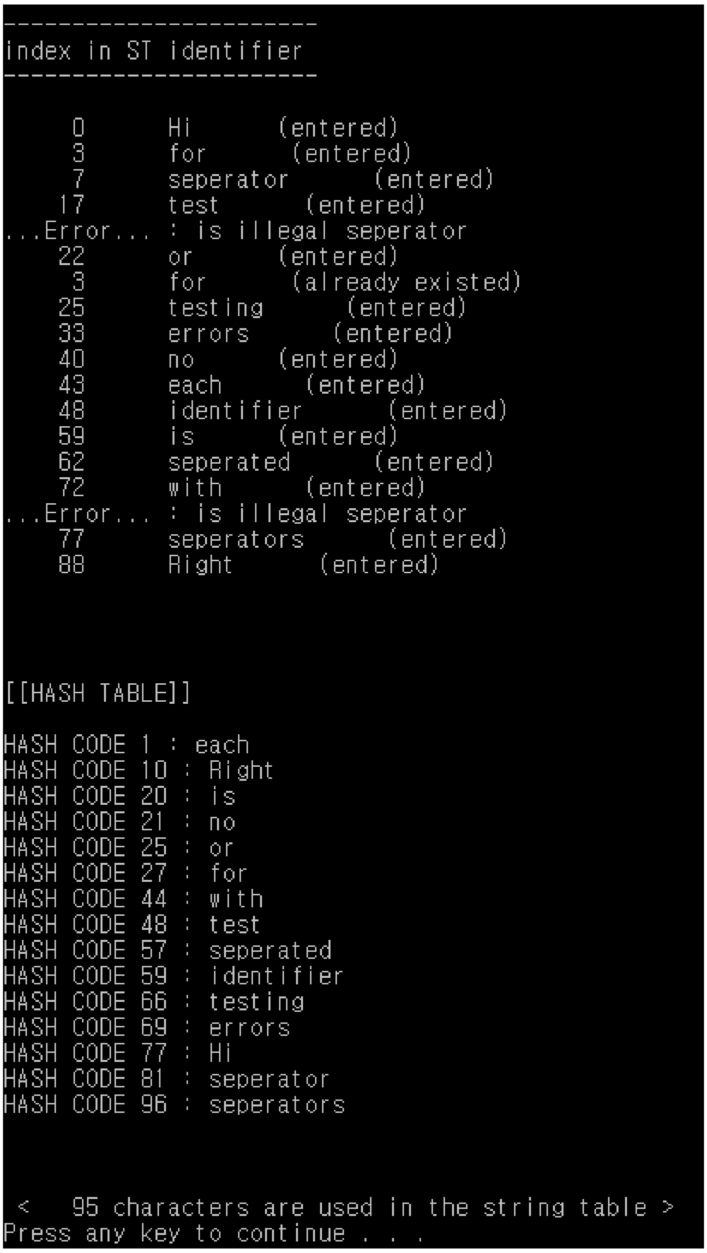


```
-----------------------
index in ST identifier
-----------------------

     0      x1      (entered)
     3      y2a34z      (entered)
    10      abc      (entered)
    14      acb      (entered)
    18      cba      (entered)
    22      cab      (entered)
    10      abc      (already existed)
    14      acb      (already existed)
    26      ab      (entered)
    29      VERY_very1      (entered)
    40      VERY_very      (entered)
...Error... & is illegal seperator
...Error... veryVERYlong          too long identifier
...Error... 99  Start with digit
...Error... 4a  Start with digit
    50      very_VERY1      (entered)



[[HASH TABLE]]

HASH CODE 24 : very_VERY1   VERY_very1
HASH CODE 69 : x1
HASH CODE 75 : VERY_very
HASH CODE 93 : y2a34z
HASH CODE 94 : cab    cba    acb    abc
HASH CODE 95 : ab



 <   61 characters are used in the string table >
Press any key to continue . . .
```

3.2 Input data_2

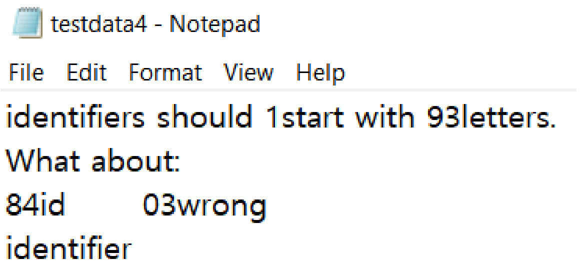


```
Hi, for seperator test:or;for testing errors?no!
each identifier.is.seperated,with:seperators;
Right?
```

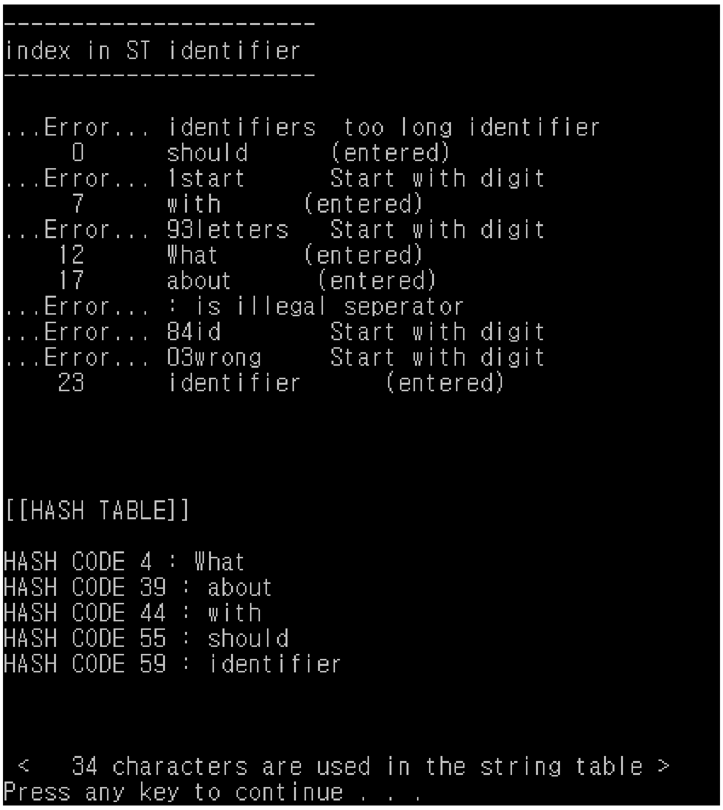


```
-----------------------
index in ST identifier
-----------------------

     0      Hi      (entered)
     3      for      (entered)
     7      seperator      (entered)
    17      test      (entered)
...Error... : is illegal seperator
    22      or      (entered)
     3      for      (already existed)
    25      testing      (entered)
    33      errors      (entered)
    40      no      (entered)
    43      each      (entered)
    48      identifier      (entered)
    59      is      (entered)
    62      seperated      (entered)
    72      with      (entered)
...Error... : is illegal seperator
    77      seperators      (entered)
    88      Right      (entered)



[[HASH TABLE]]

HASH CODE 1 : each
HASH CODE 10 : Right
HASH CODE 20 : is
HASH CODE 21 : no
HASH CODE 25 : or
HASH CODE 27 : for
HASH CODE 44 : with
HASH CODE 48 : test
HASH CODE 57 : seperated
HASH CODE 59 : identifier
HASH CODE 66 : testing
HASH CODE 69 : errors
HASH CODE 77 : Hi
HASH CODE 81 : seperator
HASH CODE 96 : seperators



 <   95 characters are used in the string table >
Press any key to continue . . .
```

3.3 Input data_3



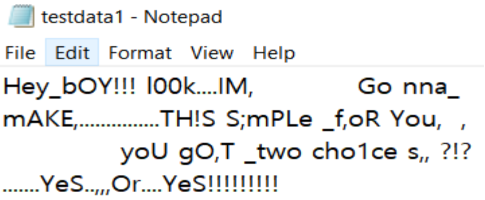
```
identifiers should 1start with 93letters.
What about:
84id       03wrong
identifier
```



```
-----------------------
index in ST identifier
-----------------------

...Error... identifiers  too long identifier
     0      should      (entered)
...Error... 1start      Start with digit
     7      with      (entered)
...Error... 93letters   Start with digit
    12      What      (entered)
    17      about      (entered)
...Error... : is illegal seperator
...Error... 84id        Start with digit
...Error... 03wrong     Start with digit
    23      identifier      (entered)



[[HASH TABLE]]

HASH CODE 4 : What
HASH CODE 39 : about
HASH CODE 44 : with
HASH CODE 55 : should
HASH CODE 59 : identifier


 <   34 characters are used in the string table >
Press any key to continue . . .
```

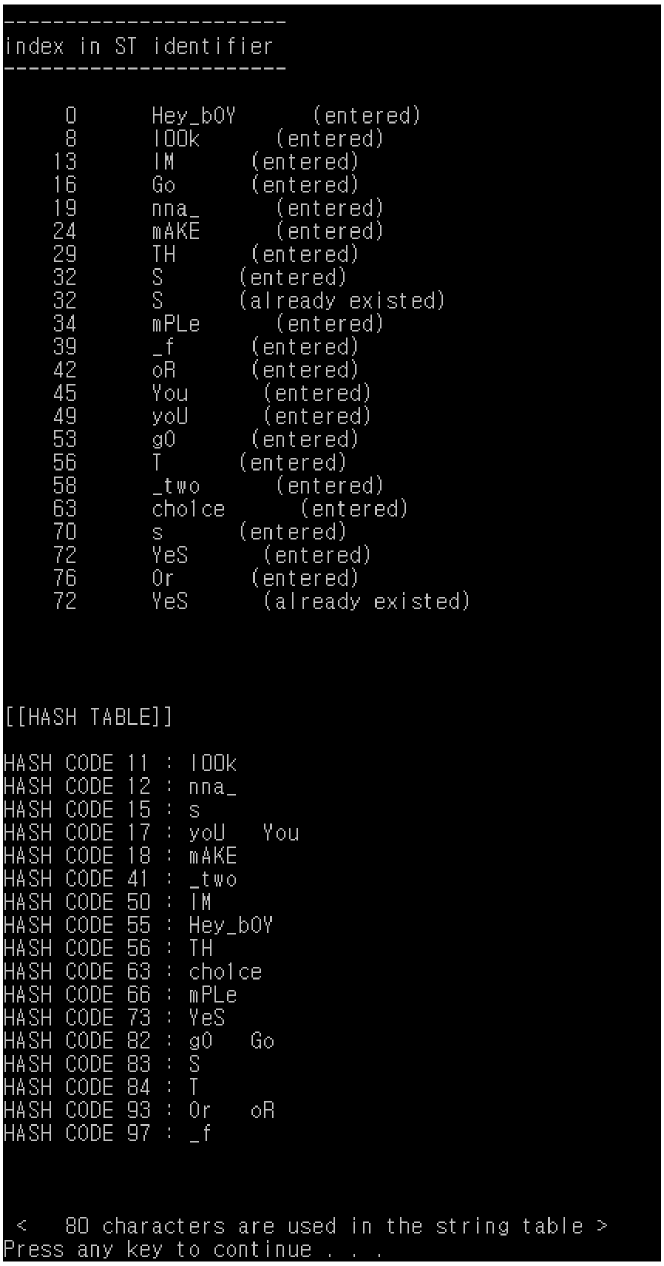
4.Data file given from professor

4.1 Input data_1



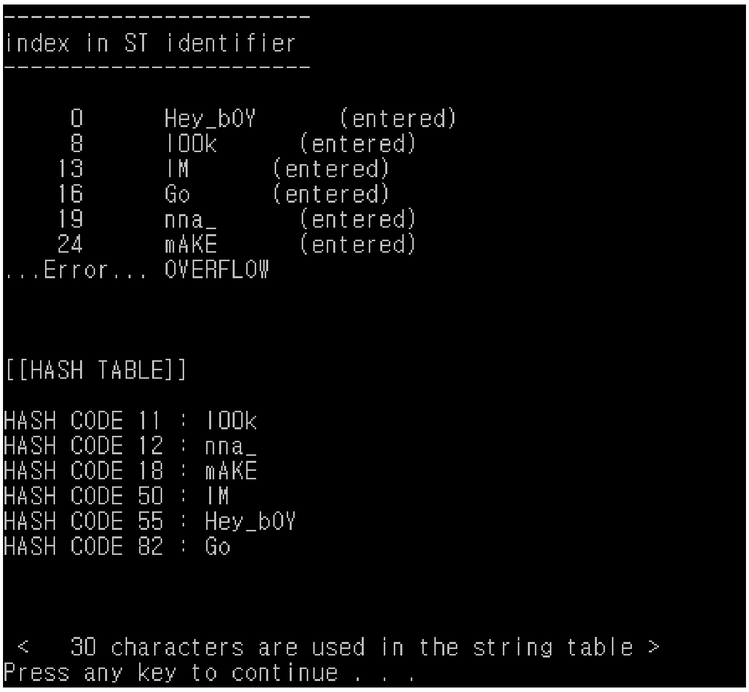
```
Hey_bOY!!! l00k....IM,            Go nna_
mAKE,...............TH!S S;mPLe _f,oR You,  ,
            yoU gO,T _two cho1ce s,, ?!?
.......YeS..,,,Or....YeS!!!!!!!!!
```

1) ST = 1000일 때



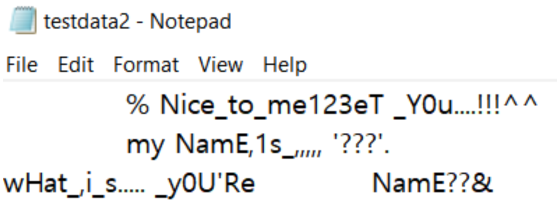
```
-----------------------
index in ST identifier
-----------------------

     0      Hey_bOY      (entered)
     8      l00k      (entered)
    13      IM      (entered)
    16      Go      (entered)
    19      nna_      (entered)
    24      mAKE      (entered)
    29      TH      (entered)
    32      S      (entered)
    32      S      (already existed)
    34      mPLe      (entered)
    39      _f      (entered)
    42      oR      (entered)
    45      You      (entered)
    49      yoU      (entered)
    53      gO      (entered)
    56      T      (entered)
    58      _two      (entered)
    63      cho1ce      (entered)
    70      s      (entered)
    72      YeS      (entered)
    76      Or      (entered)
    72      YeS      (already existed)



[[HASH TABLE]]

HASH CODE 11 : l00k
HASH CODE 12 : nna_
HASH CODE 15 : s
HASH CODE 17 : yoU   You
HASH CODE 18 : mAKE
HASH CODE 41 : _two
HASH CODE 50 : IM
HASH CODE 55 : Hey_bOY
HASH CODE 56 : TH
HASH CODE 63 : cho1ce
HASH CODE 66 : mPLe
HASH CODE 73 : YeS
HASH CODE 82 : gO   Go
HASH CODE 83 : S
HASH CODE 84 : T
HASH CODE 93 : Or   oR
HASH CODE 97 : _f



 <   80 characters are used in the string table >
Press any key to continue . . .
```

2) ST = 30일 때

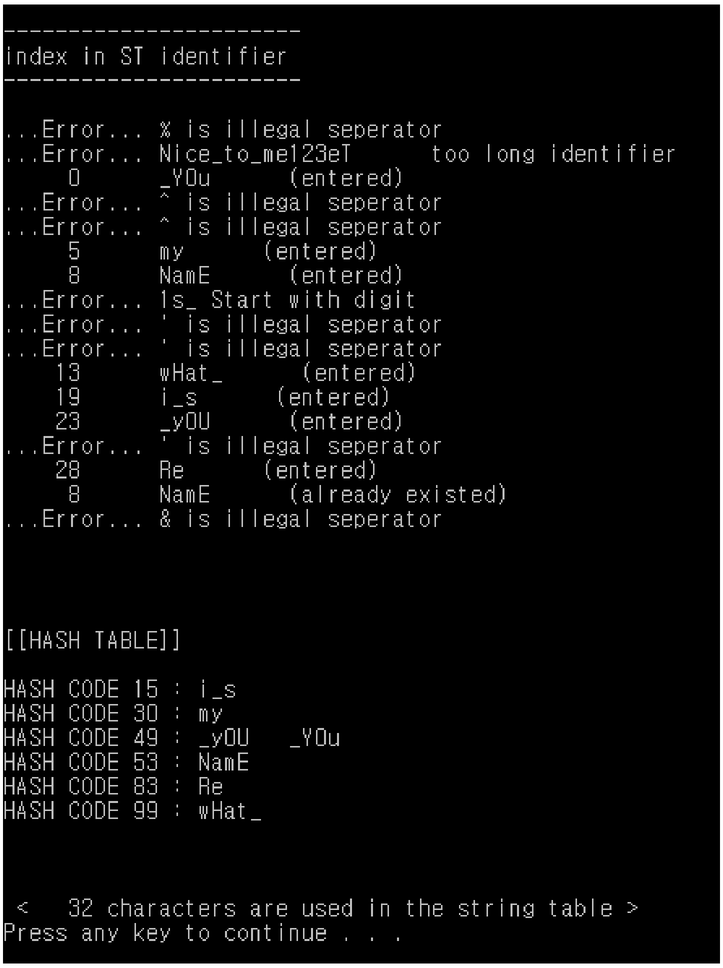


```
-----------------------
index in ST identifier
-----------------------

     0      Hey_bOY      (entered)
     8      lOOk      (entered)
    13      IM      (entered)
    16      Go      (entered)
    19      nna_      (entered)
    24      mAKE      (entered)
...Error... OVERFLOW



[[HASH TABLE]]

HASH CODE 11 : lOOk
HASH CODE 12 : nna_
HASH CODE 18 : mAKE
HASH CODE 50 : IM
HASH CODE 55 : Hey_bOY
HASH CODE 82 : Go



 <   30 characters are used in the string table >
Press any key to continue . . .
```

4.2 Input data_2

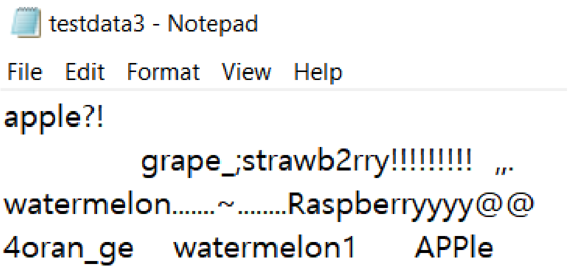


```
          % Nice_to_me123eT _Y0u....!!!^^
          my NamE,1s_,,,,, '???'.
wHat_,i_s..... _y0U'Re          NamE??&
```

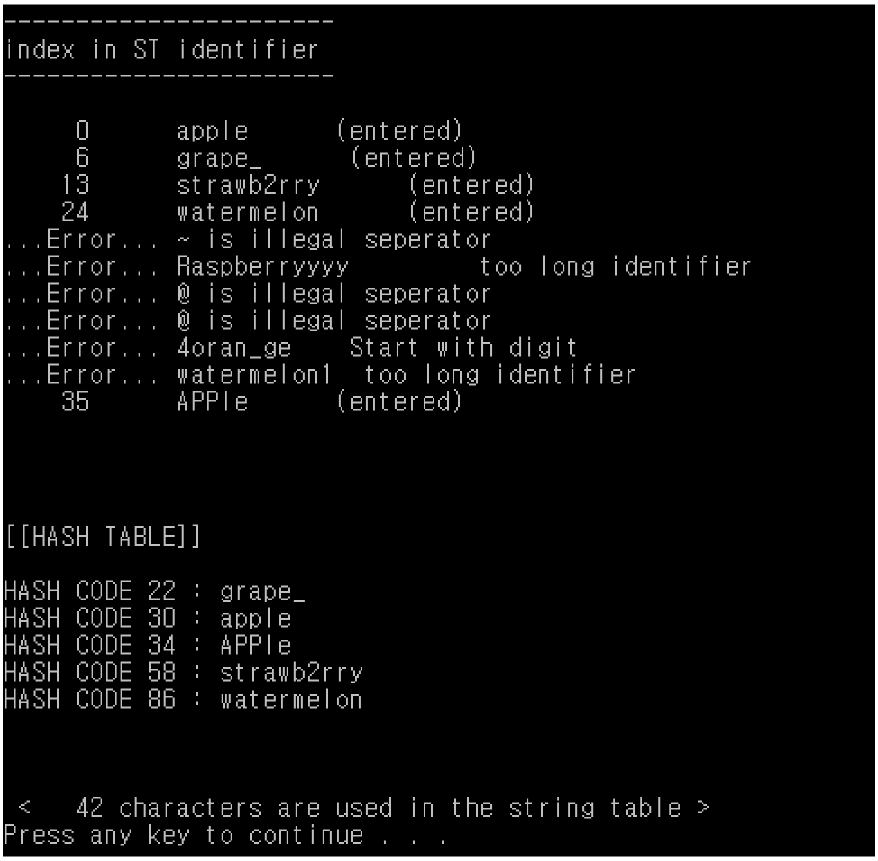


```
-----------------------
index in ST identifier
-----------------------

...Error... % is illegal seperator
...Error... Nice_to_me123eT      too long identifier
     0      _YOu      (entered)
...Error... ^ is illegal seperator
...Error... ^ is illegal seperator
     5      my      (entered)
     8      NamE      (entered)
...Error... 1s_ Start with digit
...Error... ' is illegal seperator
...Error... ' is illegal seperator
    13      wHat_      (entered)
    19      i_s      (entered)
    23      _yOU      (entered)
...Error... ' is illegal seperator
    28      Re      (entered)
     8      NamE      (already existed)
...Error... & is illegal seperator




[[HASH TABLE]]

HASH CODE 15 : i_s
HASH CODE 30 : my
HASH CODE 49 : _yOU   _YOu
HASH CODE 53 : NamE
HASH CODE 83 : Re
HASH CODE 99 : wHat_



 <   32 characters are used in the string table >
Press any key to continue . . .
```

4.3 Input data_3

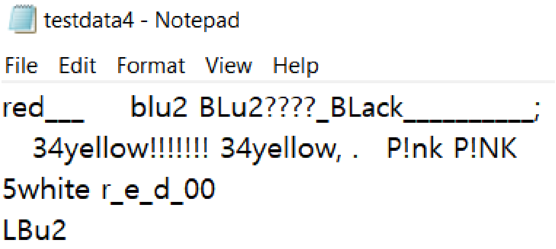


```
apple?!
              grape_;strawb2rry!!!!!!!!!  ,,.
watermelon.......~........Raspberryyyy@@
4oran_ge    watermelon1      APPle
```

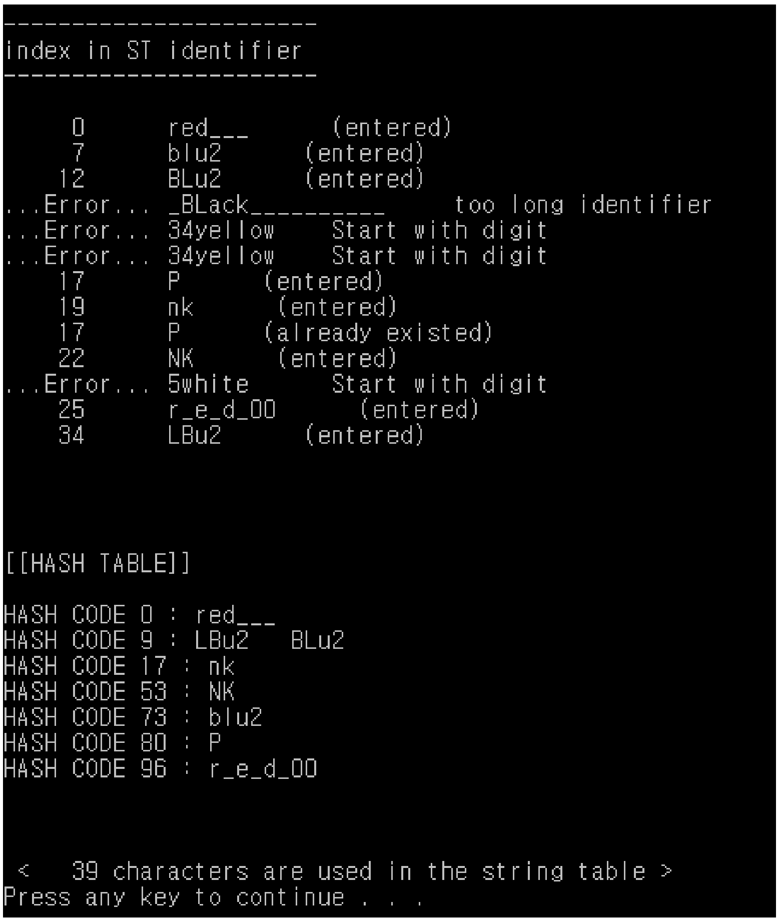


```
-----------------------
index in ST identifier
-----------------------

     0      apple      (entered)
     6      grape_      (entered)
    13      strawb2rry      (entered)
    24      watermelon      (entered)
...Error... ~ is illegal seperator
...Error... Raspberryyyy         too long identifier
...Error... @ is illegal seperator
...Error... @ is illegal seperator
...Error... 4oran_ge    Start with digit
...Error... watermelon1  too long identifier
    35      APPle      (entered)




[[HASH TABLE]]

HASH CODE 22 : grape_
HASH CODE 30 : apple
HASH CODE 34 : APPle
HASH CODE 58 : strawb2rry
HASH CODE 86 : watermelon



 <   42 characters are used in the string table >
Press any key to continue . . .
```

4.4 Input data_4



```
red___      blu2 BLu2????_BLack__________;
   34yellow!!!!!!! 34yellow, .   P!nk P!NK
5white r_e_d_00
LBu2
```



```
-----------------------
index in ST identifier
-----------------------

     0      red___      (entered)
     7      blu2      (entered)
    12      BLu2      (entered)
...Error... _BLack__________     too long identifier
...Error... 34yellow    Start with digit
...Error... 34yellow    Start with digit
    17      P      (entered)
    19      nk      (entered)
    17      P      (already existed)
    22      NK      (entered)
...Error... 5white      Start with digit
    25      r_e_d_00      (entered)
    34      LBu2      (entered)



[[HASH TABLE]]

HASH CODE 0 : red___
HASH CODE 9 : LBu2   BLu2
HASH CODE 17 : nk
HASH CODE 53 : NK
HASH CODE 73 : blu2
HASH CODE 80 : P
HASH CODE 96 : r_e_d_00



 <   39 characters are used in the string table >
Press any key to continue . . .
```