Functions with Pass-by-Reference

Lab 10: Guessing a Password

Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

11/28/2023

Purposes of the Lab

- **More on**
  - ✓ **Reference type**
  - ✓ **Pass by value & pass by reference**

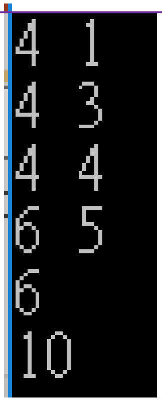
Reference Type

```
int aFunc1(int &);
int aFunc2(int &aRef);
int aFunc22(int &aRef);
void aFun3(int);
int &aFunc4(int &aRef);
int& aFunc5(int anAry[], int);
int& aFunc6(int anAry[], int &);

int main( ) {
int x =1;
int &y = x;  // Reference type should be initialized
int xx = 3;
int &zz = xx;
int anAry[] = {1, 2, 3, 4}; // declare

cout << aFunc1(x) << " " << x << endl;
// 4 1 printed
cout << aFunc2(xx) << " " << xx << endl;
// 4 4 or 4 3 printed
cout << aFunc22(xx) << " " << xx << endl;
// 4 5 or 4 4 printed
cout << aFunc3(y); // not a legal statement
cout << aFunc4(xx) << xx <<endl;
// 6 5 or 6 6 printed
cout << xx << endl; // 6 printed
cout << aFunc5(anAry, 4) << endl; //10 printed
cout << aFunc6(anAry, 4) << endl; // not a legal
statement
return 0;
}
```

Possible output



```
4 1
4 3
4 4
6 5
6
10
```

```
int aFunc1(int &a){ return a+3; }
int aFunc2(int &aRef) {return ++aRef;}
```

Increase aRef by 1 and then Return aRef

```
int aFunc22(int &aRef) {return aRef++;}
```

Return aRef and then increase aRef by 1

```
void aFunc3(int x) {x++;}
int &aFunc4(int &aRef) {aRef++; return aRef;}
```

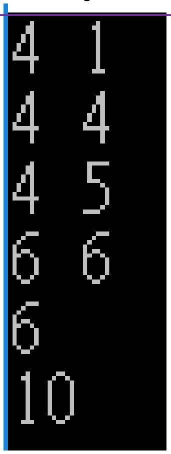
Increase aRef by 1 and then Return aRef

```
int& aFunc5(int anAry[], int inx){
anAry[2] = 10;
return anAry[2] ;}
int& aFunc6(int anAry[], int & inx) {
anAry[2] =10;
inx++;
return anAry[2]; }
```

Reference Type & cout <<

```
int aFunc1(int &);
int aFunc2(int &aRef);
int aFunc22(int &aRef);
void aFun3(int);
int &aFunc4(int &aRef);
int& aFunc5(int anAry[], int);
int& aFunc6(int anAry[], int &);
```

Output



```
4 1
4 4
4 5
6 6
6
10
```

```
int main( ) {
int x =1;
int &y = x;  // Reference type should be initialized
int xx = 3;
int &zz = xx;
int anAry[] = {1, 2, 3, 4}; // declare

cout << aFunc1(x) << " " << x << endl;
// 4 1 printed
cout << aFunc2(xx) << " "; // 4 printed
cout << xx << endl;  // 4 printed
cout << aFunc22(xx) << " "; // 4 printed
cout << xx << endl; // 5 printed
cout << aFunc4(xx) ; // 6 printed
cout << xx <<endl;  //6 printed
cout << xx << endl; // 6 printed
cout << aFunc5(anAry, 4) << endl; //10 printed
return 0;
}
```

```
int aFunc1(int &a){ return a+3; }
int aFunc2(int &aRef) {return ++aRef;}
```

Increase aRef by 1 and then Return aRef

```
int aFunc22(int &aRef) {return aRef++;}
```

Return aRef and then increase aRef by 1

```
void aFunc3(int x) {x++;}
int &aFunc4(int &aRef) {aRef++; return aRef;}
```

Increase aRef by 1 and then Return aRef

```
int& aFunc5(int anAry[], int inx){
anAry[2] = 10;
return anAry[2] ;}
int& aFunc6(int anAry[], int & inx) {
anAry[2] =10;
inx++;
return anAry[2]; }
```

Mysterious “cout <<”

int funcA();
int funcB();
int funcC();

```
cout << funcA() << funcB() << funcC();
```

may not be the same as

```
cout << funcA();
cout << funcB();
cout << funcC();
```

The fact is that funcA(), funcB(), funcC() in

```
cout << funcA() << funcB() << funcC();
```

are evaluated in an arbitrary order.

Lab 10: Guessing a Password

- Write a program that will guess a password as follows:
  - You are given a function string generatePassWd(int&) to generate a password that contains at most four symbols, for example “aBc5”. A symbol could be one of the letters, a through z, A through Z, 0 through 9. This function returns a password passWd. The parameter returns the length of the password. For example, aBc5 is returned and the length is 4.
  - Write a program to read from a keyboard a string which is a guess of a passWd you made. Print out “Too high” if the string read from the keyboard is greater than passWd or print out “Too low” if it is smaller than passWd. The symbols in a passwD are ordered in terms of the decimal values of corresponding ASCII code. That is, $0 < 1 < 2 < \cdots < 9 < A < B < \cdots < Z < a < b < \cdots < y < z$. Strings are compared in terms of their pseudo-lexicographic order based on the order given by the symbols. For example, 0<1<a<z<00<01<11<a0<zz<000<001, a < b, aa < ab, abc < abca, etc. (see next page)
  - You should continue to read strings from the keyboard until you guess the password right. That is, the password read from the keyboard is the same as passWd.

Guessing a Password

- **If a right guess is made, print "Bravo, you guess it right!". Moreover, if the number of guesses you made for a right guess is smaller than or equal to X=$\lceil \log_2 62^{len+1}/61 \rceil$+3, then print out "You know the secret!", where len is the length of a password and ⌈ ⌉ is the *ceiling* function (page 194). Otherwise, print out "You should be able to do better." Here, you should use log2() function rather than log() or log10().**

Pseudo-lexicographic order (our definition differs from that on the web):

Based on the order of the symbols defined in the previous slide, the pseudo-lexicographic order is defined as follows for two given strings $a_1 a_2 \ldots a_k$ and $b_1 b_2 \ldots b_m$:

- If $k = m$, the order of the two strings depends on the order of the symbols in the first place *i* where the two words differ (counting from the beginning of the stringss): $a_1 a_2 \ldots a_k < b_1 b_2 \ldots b_m$ if and only if $a_i < b_i$ in the underlying order of the alphabet *A*. For example, we have $hkL2H < hkM6g$.
- If $k \neq m$, the longer string is greater than the shorter string. For example, we have $0000 > zzz$.

Note that our pseudo-lexicographic differs from that on the web.

Scoring

- Your score will be calculated as follows:
- **min (100, ($1 - 0.5 * ($#ofGuess-X)/X)*100)**
    - For example, if X = 20, and #ofGuess = 22, then the score will be min(100, (1-0.5*(22-20)/20)*100) = min(100, 95) =95.

string generatePassWd(int &)

```
string generatePassWd(int &passLen){
    string tempStr=“”;
    srand(time(0));
    passLen = rand()%4+1;
    for (int i=0; i<passLen; i++){
        tempStr = tempStr + “ ”;
        int tNum= rand()%62;
        if(tNum >=0 && tNum<=9)
                tempStr[i] = ‘0' + tNum;
        else if(tNum >=10 && tNum<=35)
                tempStr[i] = ‘A' + tNum-10;
        else
                tempStr[i] = ‘a' + tNum-36;
    }
     return tempStr;
}
```

Hints & Notes

- Possible approach
  - Treat a password as a number with a base of 62 (=26+26+10). So, convert a password into a corresponding decimal value. For example, “0” =**1**, “1” = **2**, “A”=**11**, “B”=**12**, “C”=**13**, “Z”=**36**, ”a”=**37**, “b”=**38**, “z”=**62**, “00”=**63**, “01”=**64**, “10” = **125**, “zz” = **3906**, …, “zzzz” = **15018570**. That is, given a password $d_{k-1}d_{k-2}$ ... $d_1\ d_0$ of length *k*, the corresponding decimal value is $d_{k-1} \times 62^{k-1} + d_{k-2} \times 62^{k-2} + \cdots + d_1 \times 62 + d_0$. Note that there is not a zero and $0 \leq d_i \leq 62$ for each i. This will be easier for you to guess a password (automatically). *However, converting such a number back to a string (i.e., a password) is a bit tricky. If you are not able to get through it, ask me for a copy of the code.*
  - Do **binary search** to guess a password.
- Note that 1 and l look the same on the monitor.
- Note that guessing a password would be difficult if there is not a “too high” or “too low” response.

Requirements (1)

- Write a function

  status guess(string, string);

  where status is an enumeration type:

  enum status {TH, TL, RT}; // TH: too high; TL: too low; RT: right

  The function should return a guess result, TH, TL, or RT. The first parameter is a guess. The second parameter is the string we would like to guess.

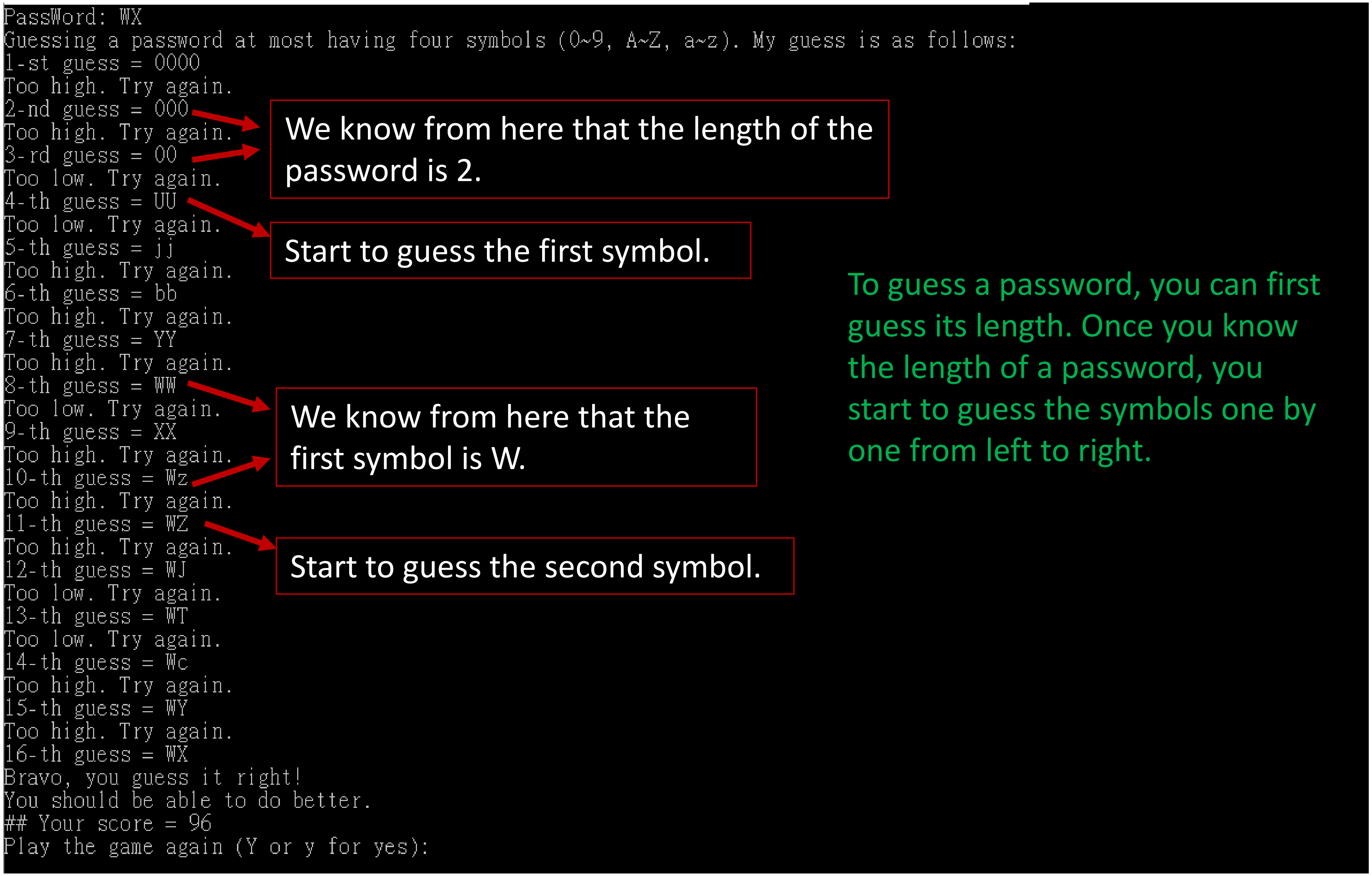
- The main() function should have calls to this function as follows:

```
Int main()
{
  string aGuess;  // A guess read from keyboard
  string passWd; // passWd we would like to guess
  int passLen;     // passWd length
          …
   guess(aGuess, passWD);
          …
}
```

Requirements (2)

- **After a right guess is made, your program should ask whether to play the game again by presenting a prompt message "Play the game again (Y or y for yes): ". Otherwise, the program terminates.**

Example of Input & Output



```
PassWord: WX
Guessing a password at most having four symbols (0~9, A~Z, a~z). My guess is as follows:
1-st guess = 0000
Too high. Try again.
2-nd guess = 000
Too high. Try again.
3-rd guess = 00
Too low. Try again.
4-th guess = UU
Too low. Try again.
5-th guess = jj
Too high. Try again.
6-th guess = bb
Too high. Try again.
7-th guess = YY
Too high. Try again.
8-th guess = WW
Too low. Try again.
9-th guess = XX
Too high. Try again.
10-th guess = Wz
Too high. Try again.
11-th guess = WZ
Too high. Try again.
12-th guess = WJ
Too low. Try again.
13-th guess = WT
Too low. Try again.
14-th guess = Wc
Too high. Try again.
15-th guess = WY
Too high. Try again.
16-th guess = WX
Bravo, you guess it right!
You should be able to do better.
## Your score = 96
Play the game again (Y or y for yes):
```

We know from here that the length of the password is 2.

Start to guess the first symbol.

We know from here that the first symbol is W.

Start to guess the second symbol.

To guess a password, you can first guess its length. Once you know the length of a password, you start to guess the symbols one by one from left to right.

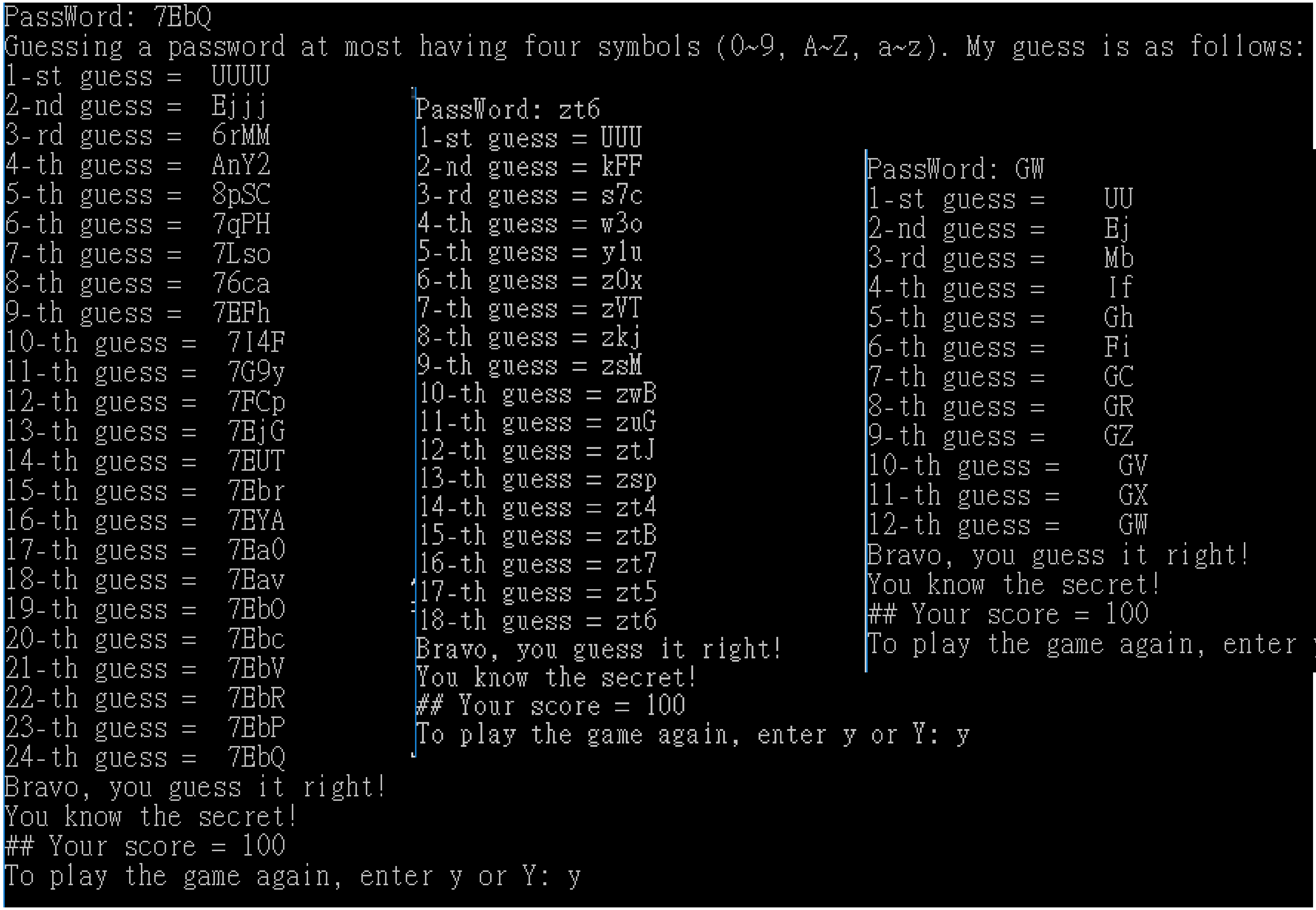
Note: You should have 1-st, 2-nd, 3-rd, 4-th, …, 10-th, 11-th, 12-th, 13-th, …, 20-th, 21-th, ….

- The above is not a good way of guessing the password.

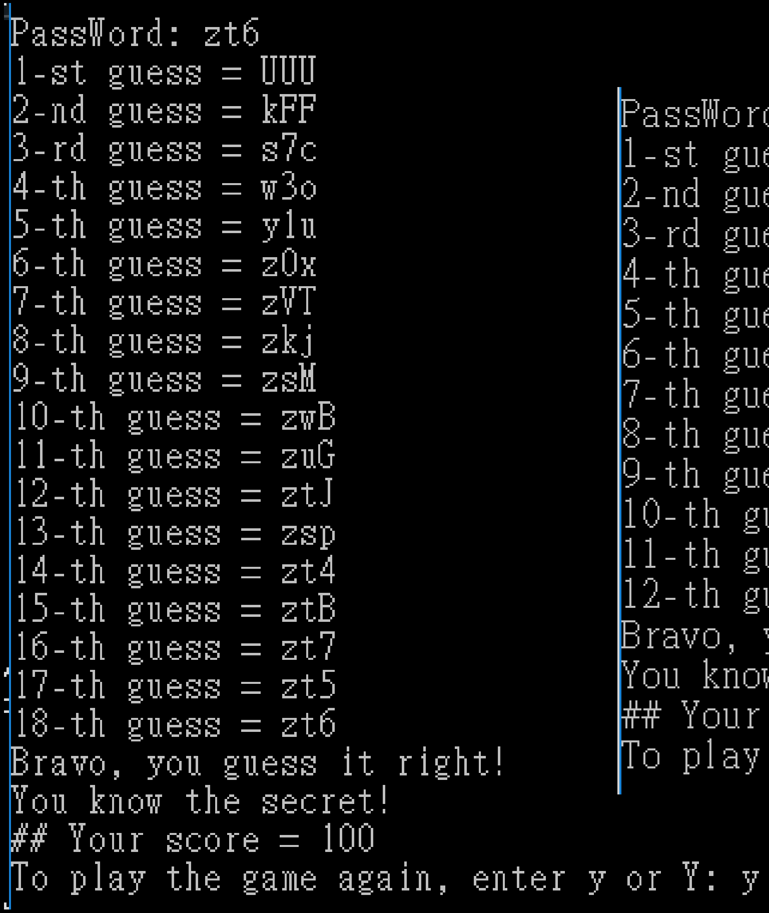
Bonus: Automatic Guessing

- A bonus of 30 points.
- Write a program to make an automatic guess. That is, repeat making a guess and receiving a response until a right guess is made.
- For this part, you can change **status guess(string, string)** freely or replace this function with another function. Certainly, you can also modify your main function.

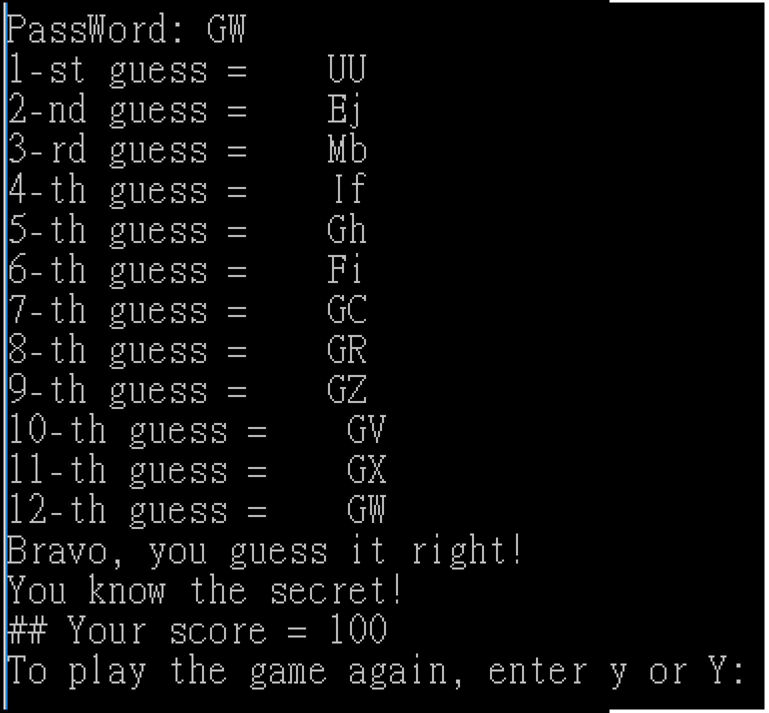
Examples of Automatic Guessing



```
PassWord: 7EbQ
Guessing a password at most having four symbols (0~9, A~Z, a~z). My guess is as follows:
1-st guess =  UUUU
2-nd guess =  Ejjj
3-rd guess =  6rMM
4-th guess =  AnY2
5-th guess =  8pSC
6-th guess =  7qPH
7-th guess =  7Lso
8-th guess =  76ca
9-th guess =  7EFh
10-th guess =  7I4F
11-th guess =  7G9y
12-th guess =  7FCp
13-th guess =  7EjG
14-th guess =  7EUT
15-th guess =  7Ebr
16-th guess =  7EYA
17-th guess =  7Ea0
18-th guess =  7Eav
19-th guess =  7Eb0
20-th guess =  7Ebc
21-th guess =  7EbV
22-th guess =  7EbR
23-th guess =  7EbP
24-th guess =  7EbQ
Bravo, you guess it right!
You know the secret!
## Your score = 100
To play the game again, enter y or Y: y
```



```
PassWord: zt6
1-st guess = UUU
2-nd guess = kFF
3-rd guess = s7c
4-th guess = w3o
5-th guess = y1u
6-th guess = z0x
7-th guess = zVT
8-th guess = zkj
9-th guess = zsM
10-th guess = zwB
11-th guess = zuG
12-th guess = ztJ
13-th guess = zsp
14-th guess = zt4
15-th guess = ztB
16-th guess = zt7
17-th guess = zt5
18-th guess = zt6
Bravo, you guess it right!
You know the secret!
## Your score = 100
To play the game again, enter y or Y: y
```



```
PassWord: GW
1-st guess =    UU
2-nd guess =    Ej
3-rd guess =    Mb
4-th guess =    If
5-th guess =    Gh
6-th guess =    Fi
7-th guess =    GC
8-th guess =    GR
9-th guess =    GZ
10-th guess =    GV
11-th guess =    GX
12-th guess =    GW
Bravo, you guess it right!
You know the secret!
## Your score = 100
To play the game again, enter y or Y:
```