제출일 : 2022. 04. 09.

KWANGWOON UNIVERSITY

담당교수	이강훈 교수님	학 번	2021203034
학 과	소프트웨어학부	이 름	허찬영

KWANGWOON UNIVERSITY

■ For each requirement, explain how you fulfilled it

Initialization

```
int main(void)
{
    int num, numPW, numSP, chance, select, Tsize, cntE=0, cntH=0;
    string newWord, ranWord, userInp, temp;
    vector<string> words = { "add", "bang", "base", "chain", "charter", "coast", "coincide", "command", "commerce", "corner",
    "craftsman", "deficiency", "deliver", "depend", "desk", "dimension", "dirty", "disaster", "drift", "duck", "economic",
    "egg", "engineering", "escape", "example", "float", "frank", "gorgeous","graceful", "great", "hightlight", "hyphothesis", "icecream",
    "jungle", "leaf","library", "mom", "null", "observer", "over", "plain","register",
        "umbrella", "unlike", "van", "veil", "venture", "visit", "vote", "wriggle"};
    vector<string> Ewords, Hwords;
    int digit[100] = { 0, };

    SaveEasyWords(words, Ewords);
    SaveHardWords(words, Hwords);

    /*for (int i = 0; i < 10; i++) //A statement to check whether a word is properly initialized
    {
```

vector를 이용하여 50개의 영어 단어를 초기화 했다.

List management

```
void PrintWords(vector<string>& words) //print words initialized
{
    cout << "A total of " << words.size() <<" words are available : \n";
    sort(words.begin(), words.end());
    for (const auto& word:words) {
        cout << word <<'\n';
    }cout << '\n';
```

```
Choose an option (1-3): 1

A total of 50 words are available :
add
bang
base
chain
charter
coast
coincide
command
commerce
corner
craftsman
deficiency
deliver
depend
desk
dimension
dirty
disaster
drift
duck
economic
```

sort() 함수를 사용하여 words에 있는 영어 단어를 사전적 순서로 정렬했다.

```
Choose an option (1-3): 2

Input a new word: programming
The word "programming" has been succesfully inserted in the list.

[  MANAGE LIST  ]
1. Print Words
2. Add word
3. Return

Choose an option (1-3): 2

Input a new word: add
The word "add" already exists in the list.
```

```
string AddWord(vector<string>& words) //Add a new word in vector
{
    string newWord;
    int overlap = 0;
    cout << "Input a new word: ";
    cin >> newWord;
    if (!cin) {
        cout << "Failed to add new word\n";
        return "ERROR!";
    }
    else {
        for (int i = 0; i < words.size(); i++) {
            if (newWord == words[i])
                overlap = 1;
        }
        if (overlap == 1) {
            cout << "The word \"" << newWord << "\" already exists in the list.\n\n";
            return "ERROR!";
        }
        else
            cout << "The word \"" << newWord << "\" has been succesfully inserted in the list.\n\n";
    }

    return newWord;
}
```

AddWord 함수는 새로운 단어를 추가하는 기능을 한다. 이 함수는 words를 매개변수로 받고 새로운 단어(newWord)를 반환하는 함수다. 하지만 만약 사용자가 기존에 있는 단어를 입력할 경우 ERROR! 문자를 반환하게 된다.

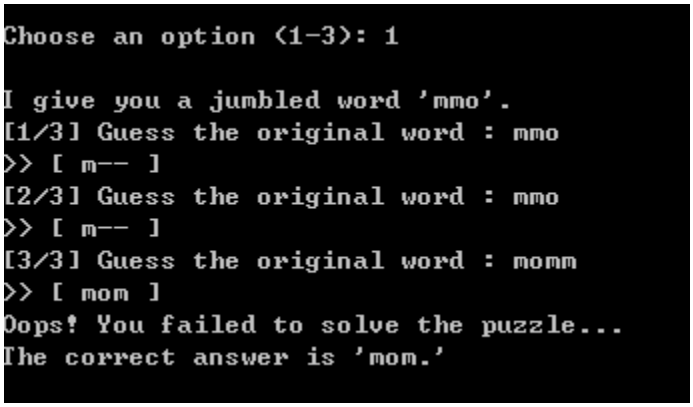
■ Puzzle solving

Choose the difficult (easy, hard)

```
vector<string>SaveEasyWords(vector<string>& words, vector<string>& Ewords)
{
    for (int i = 0; i < words.size(); i++) {
        if (words[i].length() <= 5) {
            Ewords.push_back(words[i]);
        }
    }
    return Ewords;
}
```

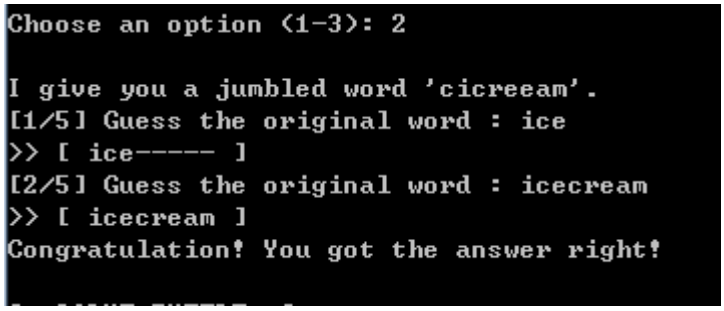
```
int SelRanEWord(vector<string> Ewords)
{
    int select;
    srand((unsigned int)time(NULL));
    select = rand() % Ewords.size();
    return select;
}

string SwapEWords(vector<string> Ewords, int select) //a function randomly se
{
    int swapNum1, swapNum2, swapNum3, swapNum4;
    srand((unsigned int)time(NULL));
    cout << "I give you a jumbled word '";
    swapNum1 = rand() % Ewords[select].size();
    swapNum2 = rand() % Ewords[select].size();
    swapNum3 = rand() % Ewords[select].size();
    swapNum4 = rand() % Ewords[select].size();

    swap(Ewords[select][swapNum1], Ewords[select][swapNum2]);
    swap(Ewords[select][swapNum3], Ewords[select][swapNum4]);
    cout << Ewords[select] << "'.\n";
    //cout << "Guess the original word : "<< Ewords[select] <<'\n';

    return Ewords[select];
}
```

우선, SaveEasyWord 함수는 words에 있는 단어 중 길이가 5 이하인 단어를 Ewords에 담는 기능을 한다. 그리고 SelRanEword 함수는 rand() 함수를 이용하여 정수를 랜덤으로 반환하는 기능을 한다. 마지막으로 SwapEwords 함수는 사용자에게 출력될 단어를 뒤섞는 역할을 한다. SelRanEword 함수에서 반환된 정수를 이용하여 Ewords의 단어 중 하나를 무작위로 선택하고 sawp() 함수를 이용하여 사용자에게 출력될 단어를 뒤섞었다. (문제에서는 swapping을 두 번 하라고 지시했지만 그렇게 될 경우 게임의 난이도가 너무 쉬워져서 네 번으로 설정했다.)



main 함수에 있는 chance 변수를 이용하여 chance의 값이 3을 넘어갈 경우 while문을 break하여 특정 메시지와 함께 진행 중인 게임을 종료 시키도록 구현했다.

Hard 난이도 역시 Easy 난이도와 똑같은 방식으로 코드를 작성하였고 Easy 난이도와 다른 점은 단어의 길이가 5를 넘을 경우 Hword에 단어를 저장하고 사용자에게 생각할 기회를 다섯 번 주었다. 그리고 게임의 난이도를 조금 더 올리기 위해 Hard 난이도는 Swapping을 6번하도록 설정했다. (문제에서는 네 번 하도록 지시하였다)



사용자가 주어진 기회 안에 정답을 입력할 경우 축하 메시지와 함께 통과하도록 코드를 작성했다.

```
string CheckProblem(string word, string user, int digit[])
{
    string temp;

    if (user.size() >= word.size())
    {
        temp = user;
        //temp.erase(word.size(), user.size());
        for (int i = 0; i < word.size(); i++)
        {
            if (digit[i] == 0) {
                if (user[i] == word[i]) {
                    temp[i] = word[i];
                }
                else
                    temp[i] = '-';
            }
            else {
                temp[i] = word[i];
            }
        }

    }

    else if (user.size() < word.size()) {
        for (int i = 0; i < word.size(); ++i) {
            temp.insert(i, "-");
        }
        for (int i = user.size(); i < word.size(); ++i) {
            user.insert(i, "-");
        }
        for (int i = 0; i < word.size(); i++)
        {
            if (digit[i] == 0) {
                if (user[i] == word[i]) {
                    temp[i] = word[i];
                }
                else
                    temp[i] = '-';
            }
            else {
                temp[i] = word[i];
            }
        }
    }

    return temp;
}
```

CheckProblem 함수는 무작위로 선택된 단어(word)와 사용자가 입력 받은 단어(user), 그리고 int형 배열 digit를 매개 변수로 하여 사용자가 맞춘 스펠링(temp)을 반환하는 함수이다. (맞추지 못한 스펠링은 '-'로 설정한다) 이 함수를 통해 사용자가 어느 부분에서 똑같은 스펠링을 입력했는지 체크하고 확인할 수 있다. digit은 사용자가 맞춘 스펠링의 index를 확인하기 위해 선언된 배열로 초기에 0으로 초기화 되어 있지만 사용자가 올바른 스펠링을 입력할 경우 해당 index값에 1을 저장한다. CheckProblem에 반환된 값을 이용하여 정답/오답 유무를 확인할 수 있다.

Repetitive execution

전체적으로 while문을 이용하여 특정 정수가 입력될 때까지 프로그램이 반복되도록 설정했다.

Requirements(implementation)

```
void PrintMessage1();
void PrintMessage2();
void PrintMessage3();
int ChooseAnOption(void);
void PrintWords(vector<string>&);
string AddWord(vector<string>&);
vector<string> SaveEasyWords(vector<string>&, vector<string>&);
vector<string> SaveHardWords(vector<string>&, vector<string>&);
int SelRanEWord(vector<string>);
string SwapEWords(vector<string>, int);
int SelRanHWord(vector<string>);
string SwapHWords(vector<string>, int);
string CheckProblem(string, string, int*);
```

전체적으로 vector와 string을 이용하였다. digit의 경우에는 배열을 사용하여 관리하는 것이 조금 더 나을 것 같아서 배열을 사용하였다. 그리고 다양한 함수를 사용하여 코드를 작성했다.

```
void PrintWords(vector<string>& words) //print words initialized
{
    cout << "A total of " << words.size() << " words are available : \n";
    sort(words.begin(), words.end());
    for (const auto& word : words) {
        cout << word << '\n';
    }cout << '\n';
}
```

Range-based for문을 사용하여 주어진 단어를 모두 출력하는 기능을 구현했다.

```
int SelRanEWord(vector<string> Ewords)
{
    int select;
    srand((unsigned int)time(NULL));
    select = rand() % Ewords.size();
    return select;
}
```

문제에서는 randint() 함수를 통해 난수를 발생하도록 지시했지만 randint()함수를 사용할 경우

약간의 문제가 발생하였다.

```
/*int SelRanEWord(vector<string> Ewords)
{
    int select;
    select = randint(0, Ewords.size())
    return select;
}*/
```

randint() 함수를 사용하려면 위 사진처럼 코드를 작성하면 된다. 하지만 randint() 함수를 사용하면 지정된 난수만 출력되어 동일한 패턴으로 단어가 출력되었다. 그렇기 때문에 게임의 난이도가 쉬워졌고 완성도가 떨어졌다. 그래서 randint() 함수 대신에 rand() 함수를 사용하여 완전한 무작위 수가 나오도록 코드를 작성했다.

(sort(), swap()는 앞에서 설명했다.)

■ Demonstrate the correctness of your code

Overall control flow

```
while (true)
{
    PrintMessage1();
    num = ChooseAnOption();
    switch (num) {
    case 1:
        while (true) {
            PrintMessage2();
            numPW = ChooseAnOption();

            switch (numPW) {
            case 1:
                PrintWords(words);
                break;
            case 2:
                newWord = AddWord(words);
                if (newWord != "ERROR!")
                    words.push_back(newWord);
                if (newWord.length() <= 5)
                    Ewords.push_back(newWord);
                else if (newWord.length() > 5)
                    Hwords.push_back(newWord);

                break;
            case 3:
                break;
            }
            if (numPW == 3)
                break;
        }
        break;
    case 2: //GAME ALGORITHM
        while (true) {
```

```
<<< WORD JUMBLE >>>
1. Manage List
2. Solve Puzzle
3. Exit

Choose an option (1-3): 2

[  SOLVE PUZZLE  ]
1. Easy Mode
2. Hard Mode
3. Return

Choose an option (1-3): 1

I give you a jumbled word 'bgna'.
[1/3] Guess the original word : ba
>> [ ba-- ]
[2/3] Guess the original word : ban
>> [ ban- ]
[3/3] Guess the original word : bang
>> [ bang ]
Congratulation! You got the answer right!

[  SOLVE PUZZLE  ]
1. Easy Mode
```

전체적인 게임의 흐름은 main 함수 안에서 작성되었다. 기본적으로 while문을 사용하여 특정한 정수가 입력될 때까지 무한 반복 시키도록 구현하였고 switch-case문을 이용하여 사용자가 입력한 수에 따라서 특정 기능을 하도록 설정했다.

Menu input and output (out-of range input)

```
<<< WORD JUMBLE >>>
1. Manage List
2. Solve Puzzle
3. Exit

Choose an option (1-3):
```

```
[  MANAGE LIST  ]
1. Print Words
2. Add word
3. Return

Choose an option (1-3):
```

<<< WORD JUMBLE >>>은 이 프로그램의 메인 화면이며 1번을 선택할 경우 리스트를 관리할 수 있다. 리스트 관리 화면에서 1번을 입력할 경우 모든 단어가 출력되고 2번을 출력하면 새로운 단어를 추가할 수 있다. 3번을 입력하면 메인 화면을 돌아간다.

```
[  SOLVE PUZZLE  ]
1. Easy Mode
2. Hard Mode
3. Return

Choose an option (1-3):
```

메인 화면에서 2번을 입력할 경우 게임의 난이도를 설정할 수 있다. 1번을 입력할 경우 쉬운 난이도로 게임이 진행되고 2번을 입력할 경우 어려운 난이도로 게임이 진행된다. 3번을 입력하면 메인 화면으로 돌아간다.

```
<<< WORD JUMBLE >>>
1. Manage List
2. Solve Puzzle
3. Exit

Choose an option (1-3): 3

Good Bye!!
```

마지막으로 메인 화면에서 3번을 입력할 경우 메시지와 함께 프로그램이 종료된다.

```
int ChooseAnOption(void)
{
    int num;
    while (true) {
        cout << "Choose an option (1-3): ";
        cin >> num;
        if ((num >= 1) && (num <= 3)) {
            cout << '\n';
            break;
        }
    }
    return num;
}
```

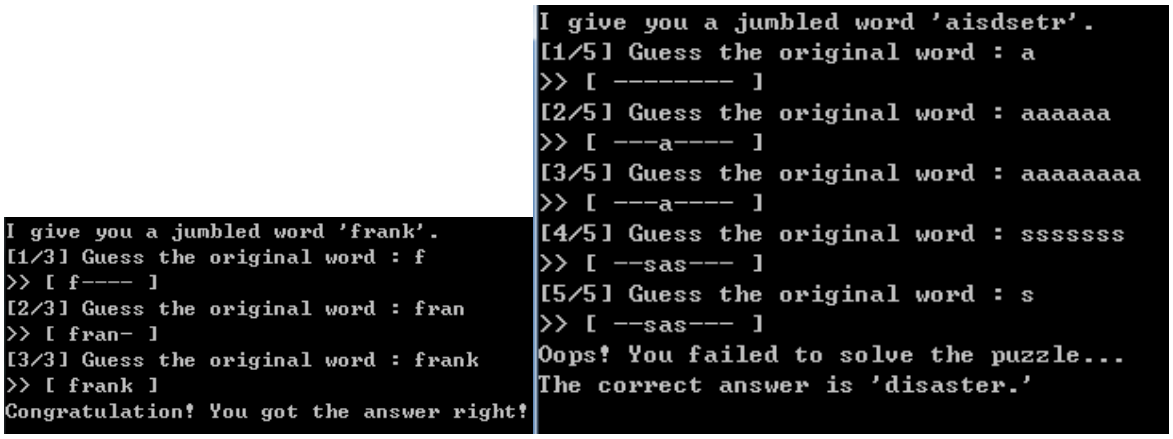
ChooseAnOption 함수를 사용하여 간단한 메시지를 출력하였고 while문과 if문을 활용하여 사용자가 특정 정수 외에 다른 문자를 입력하는 경우를 방지하였다.

List print and add (duplicate word)

Manage List에서 Print words를 선택할 경우 ranged-based for문을 사용하여 목록과 단어의 개수를 출력하였고 Add word를 선택할 경우 AddWord 함수를 통해 새로운 단어를 입력 받았다. (자세한 코드 설명은 위에 Addword 함수와 ranged-based for문 참고)

Correct selection and jumbling of word (easy vs. hard mode)

SaveEasyWords, SaveHardWords, SelRanEWord, SelRanHWord, SwapEWords, SwapHWords 함수를 이용하여 난이도를 설정하고 무작위로 단어를 선택하여 뒤섞은 후 사용자에게 출력했다.

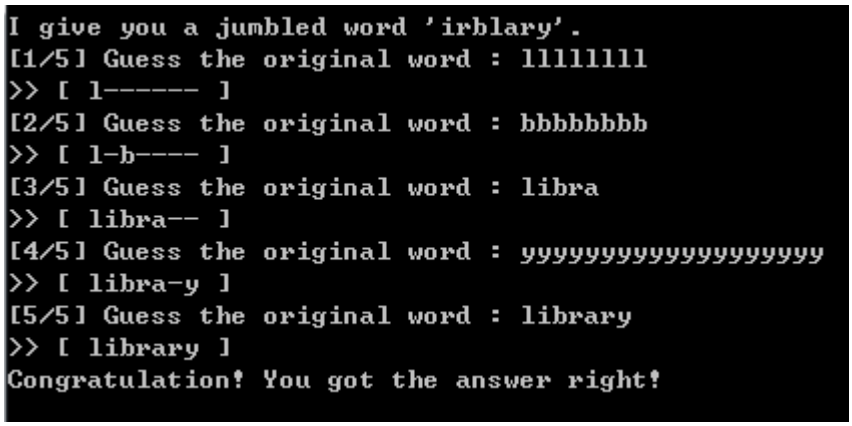


쉬운 난이도의 경우 5개 이하의 단어가 선택되고 세 번의 기회가 주어진다.

어려운 난이도는 6개 이상의 단어가 선택되고 다섯 번의 기회가 주어진다.

(코드의 자세한 설명은 함수 부분 설명 참고)

Exact checking of correct parts of the guessed word



이 게임은 사용자가 입력한 문자가 선택된 단어와 동일한 경우 통과하는 방식으로 진행된다. 사용자가 단어를 조금 더 쉽게 추측할 수 있도록 단어 전체의 스펠링이 맞지 않더라도 맞는 부분의 스펠링을 확인 시켜주는 기능을 구현했다. 사용자가 더 많은 길이의 단어를 입력하거나 더 짧은 단어를 입력해도 오류 없이 정상적으로 작동한다.

(코드의 자세한 설명은 CheckProblem 함수 부분 설명 참고)

■ Conclude with some comments on your work

Key challenges you have successfully tackled

프로그램을 제작하면서 오류가 굉장히 많았다. 사소한 오류부터 시작해서 코드를 전반적으로 수정해야 되는 오류까지 아주 많았다. 특히, Exact checking of correct parts of the guessed word 부분을 고치기 위해 기존에 작성했던 코드 전부를 삭제하고 다시 작성하면서까지 오류를 수정했다. 많은 오류 중에 한 가지 예를 들자면 사용자가 답에 근접한 값을 입력할 경우 오류가 발생하는 상황이 있었다.

```
I give you a jumbled word 'ncoioemc'.
[1/5] Guess the original word : economicc
>> [ economic ]
[2/5] Guess the original word : eeeeeeee
>> [ economic ]
Congratulation! You got the answer right!
```

문제에서 주어진 단어는 economic이다. 사용자가 economicc처럼 "정답 + 다른 단어"의 형태를 입력하면 오답으로 처리가 되지만 다음 번에 단어의 길이만 맞으면(economic의 길이는 8이므로 길이가 8인 단어를 아무거나 입력할 경우) 정답으로 처리 되는 오류가 발생하였다. 이 오류를 고치기 위해 원인을 분석한 결과 오류의 원인이 배열 digit을 이용하면서부터 생긴 오류라는 것을 알 수 있었다. CheckProblem 함수에서 사용자가 맞춘 부분을 더 원활하게 출력시키기 위해 digit을 사용했다. 사용자가 맞춘 부분이 있을 경우 해당 index의 digit의 값을 1로 변경하도록 했는데 이 함수에서 digit의 값이 1일 경우 스펠링을 검사하지 않고 temp에 올바른 값을 저장했다. (digit의 값이 0일 경우에는 스펠링을 검사한 뒤 맞으면 올바른 값, 틀리면 "-"을 저장함) 따라서 지난 기회에서 올바른 값을 입력한 부분은 다음 기회에서 잘못된 값을 저장해도 정답으로 출력됐다. 이 오류를 해결하기 위해 전반적인 코드를 수정해보고 사용자가 입력할 수 있는 문자에 제한을 두는 등 다양한 시도를 해봤다. 하지만 노력에 비해 오류를 수정하기 위한 코드는 매우 간단했다.

```
for (int i = 0; i < word.size(); i++)
{
    if (digit[i] == 0) {
        if (user[i] == word[i]) {
            temp[i] = word[i];
        }
        else
            temp[i] = '-';
    }
    else {
            temp[i] = word[i];
            if (user[i] != word[i]) {
                err++;
            }
    }
}
```

```
string Etemp = temp + '?';
if (err > 0) return Etemp;
else    return temp;
```

완벽한 방법은 아니지만 digit이 1일 때도 간단한 검사를 하여 사용자에게 맞췄던 부분은 출력하되 정답으로 인식이 안되도록 설정을 하였다. 만약 지난 번 기회에서 맞췄던 스펠링이 이번 기회에서 맞추지 못할 경우 오류를 감지하는 변수의 값을 변경하고 이 변수가 0보다 크다면 오답을 반환하도록 코드를 작성했다. 매우 간단하게 수정할 수 있었지만(완벽하진 않지만) 제일 고치기 어려운 오류였다. 그래도 에러의 대부분을 잘 수정하여 나름 프로그램이 잘 돌아갈 수 있던 것 같다.

Limitations you hope to address in the future

새로운 기능을 추가하고 에러를 고치기 위해 코드에 다른 코드를 덧붙이면서 코드가 매우 더러워지고 복잡 해졌다. 그렇기 때문에 코드를 조금 더 깔끔하게 고치고 싶다. 그리고 더 많고 다양한 방법의 예외 처리를 이용해서 사용자의 실수로 발생할 수 있는 오류를 줄이고 싶다.