/.

```c
#include <stdio.h>

char words[5757][5];
char data[100];

void file_r()
{
    FILE *f;
    f = fopen( filename: "words.dat",  mode: "r");
    for (int i = 0; i < 5762; i++) {
        fgets(data, sizeof(data), f);
        if(data[0] == '*')
        {
            continue;
        }
        for (int j = 0; j < 5; j++) {
            words[i - 4][j] = data[j];
        }
    }
    fclose(f);
}
int main()
{
    file_r();
    int degree = 0;
    char str[5];
    scanf("%s", str);

    for(int i = 0; i < 5757; i++)
    {
        int cnt = 0;
        for(int j = 0; j < 5; j++)
        {
            if(str[j] == words[i][j])
            {
                cnt++;
            }
        }
        if(cnt == 4)
        {
            char temp[5];
            for(int j = 0; j < 5; j++)
            {
                temp[j] = words[i][j];
            }
            printf("adjacent인 단어 = %s\n", temp);
            degree++;
        }
    }
    printf("%s의 degree는 : %d\n", str, degree);
    return 0;
}
```

```
hello
adjacent인 단어 = cello
adjacent인 단어 = hallo
adjacent인 단어 = hells
adjacent인 단어 = hullo
adjacent인 단어 = jello
hello의 degree는 : 5
```

```
graph
adjacent인 단어 = grape
adjacent인 단어 = grapy
graph의 degree는 : 2
```

words.dat 파일의 데이터를 file_r 함수를 이용하여 모두 이차원 배열 words에 저장하고 hello와 graph의 한글과 한글라 데이터와 비교하여 같은경우 cnt값을 1씩증가시켰을때 본복코이 끝났을때 4가된다면 adjacent임을 알수있다. 이를 통해 hello의 adjacent인 단어는 cello, hallo, hells, hullo jello로 degree는 5 graph는 grape, grapy로 degree는 2이다.

2~n번

```
struct record
{
    char rdata[5];
    struct record* next;
};

struct table
{
    int idx;
    struct record *emt;
};
```

2~5번을 해결하기 위해서 구조체를 4정했다. table 구조체와 record 구조체를 정의하여 Linked List 개념을 응용하였다. record는 노드의 정보 table의 idx는 같은 값의 degree를 가리는 데이터 들의 개수를 저장 하고 emt는 가장최근에 들어온 graph 데이터가 담겨있다.

```c
int main()
{
    file_r();
    struct table *table_arr = (struct table *)malloc( size: sizeof(struct table) * 5757);

    for(int i = 0; i < 5757; i++)
    {
        table_arr[i].idx = 0;
        table_arr[i].emt = NULL;
    }
    for(int i = 0; i < 5757; i++)
    {
        struct record *q = malloc(sizeof(struct record));
        for(int j = 0; j < 5; j++)
        {
            q->rdata[j] = words[i][j];
        }
        q->next = NULL;
        int degree = 0;
        for(int k = 0; k < 5757; k++)
        {
            int cnt = 0;
            for(int x = 0; x < 5; x++)
            {
                if(q->rdata[x] == words[k][x])
                {
                    cnt++;
                }
            }
            if(cnt == 4)
            {
                degree++;
            }
        }
        table_arr[degree].idx++;
        struct record *r = table_arr[degree].emt;
        if(r == NULL)
        {
            table_arr[degree].emt = q;
        }
        else
        {
            q->next = r;
            table_arr[degree].emt = q;
        }
    }
    int num = 0;
    int max = 0;
    printf("2번) 각 degree별 graph분포\n");
    for(int i = 0; i < 5757; i++)
    {
        if(table_arr[i].idx != 0)
        {
            max = i;
            printf("%d : %d\n", i, table_arr[i].idx);
            num += table_arr[i].idx;
        }
    }
    printf("3번)degree의 maximum값은? : %d\n", max);
    struct record *p = table_arr[max].emt;
    printf("4번)degree가 최대값 %d인 단어는? : ", max);
    while(p != NULL)
    {
        printf("%s ", p->rdata);
        p = p->next;
    }
    printf("\n");
    int avg = 0;
    for(int i = 0; i < 5757; i++)
    {
        if(table_arr[i].idx != 0)
        {
            avg += table_arr[i].idx * i;
        }

    }
    printf("5번)평균 degree는? : %lf\n", (double)avg / 5757);
    return 0;
}
```

table_arr이라는 구조체 배열을 만들어 0~5757의 degree를 저장할 공간을 만들고 1번과 같은 방식으로 table_arr의 각각의 idx 값을 측정해나가고 degree별 리스트도 구현했다.

```
2번) 각 degree별 graph분포
0 : 671
1 : 774
2 : 727
3 : 638
4 : 523
5 : 428
6 : 329
7 : 280
8 : 249
9 : 213
10 : 188
11 : 162
12 : 120
13 : 116
14 : 102
15 : 75
16 : 53
17 : 32
18 : 32
19 : 20
20 : 8
21 : 6
22 : 4
23 : 2
24 : 3
25 : 2
```

```
3번)degree의 maximum값은? : 25
4번)degree가 최대값 25인 단어는? : cores bares
5번)평균 degree는? : 4.910544
```

2번) 결과로 봤을때 degree는 0~25까지 있음을 알수있다.
3번) 결과를 통해 degree의 최대값은 25 이다.
4번) 25의 degree를 가지는 단어는 cores와 bares 가 있다.
5번) degree의 평균은 4.910544로 약 5 임을 알수있다.
6번) adjacency 하다는 길은 degree가 최소 1이상 임을 의미하므로 전체 데이터 - degree가 0인 데이터
= 5757 - 671 = 5086
따라서 5086 개의 adjacency 한 노드가 있다.
7번) 6번의 결과를 통해 backend.c 파일의 최소한의 Pool_SIZE는 5086이다.

이를 통해 graph List의 크기를 구해보면 전체 데이터는 5757개로 포인터 주소의 크기는 8byte 평균 degree는 5 node의 크기는 5byte(data) + 8byte(Link)를 통해
5757 * 8 + 5757 * 5 * 13 = 420261 byte가 필요하다.
420261 byte는 약 0.42MB 로 만약 matrix 로 구현했을시 크기 5757x5757 byte = 33143049
= 약 33MB 보다 약 1/100 배의 메모리 공간을 차지한다.
이것으로 List로 구현 했을때 데이터의 크기는 matrix보다 작지만 List로 구현을 해서 데이터를 탐색하기에는 시간이 오래 걸릴 것같다.