과제 2: Coalesced chain을 이용한 hashing 개발

• 과제 내용: Coalesced chain을 이용하는 해시테이블을 구현하고 실험한다.

- 삽입할 데이터는 제공되는 파일 "Companies_Korean.txt"의 각 줄에 있는 회사 이름이다. 파일 저장시에는 각 줄의 회사의 이름마다 하나의 레코드를 준비하여 프로그램에 저장한다. 레코드가 가지는 필드는 다음과 같다. (괄호속은 필드의 타입 선언)

Name : 화일에서 읽은 이름 (char name[100]) Monincome : random number 생성기를 이용하여 생성한 월간수입 (int 범위 : 0원 ~ 50,000,000원). Link : Coalesced 체인을 구성하는 링크 (int link)

결국 위 3개의 필드를 가지는 다음과 같은 구조체로 레코드를 구현한다:

```
typedef struct str_ty_record {
  char name [100];
  int monincome;
  int link;
} type_record;
```

- 레코드의 key는 name 필드를 이용한다.

- 총 레코드 수 (hash table size)는 다음과 같이 매크로로 정의한다.

```
#define Tbl_Size 37533 // 테이블의 크기 (레코드 수). 총 저장될 레코드 수의 110% 정도.
```

- 해시테이블 선언

```
yype_record Hashtable[Tbl_size ];
```

- Home address 를 구하는 hash function 에 이용되는 함수는 다음을 이용하자.
  여기와 다른 hash 함수를 사용하여도 무방하다. 단, 0 ~ (Tbl_Size – 1) 사이의 수가 골고루 나오도록 설계되어야 한다. 즉 입력문자열에 대하여 거의 random 하게 보이는 수가 출력되게 하는 것이 좋다.

Hash function

```
int hash(char recname[]) {
    unsigned char u; int HA, j, leng, halfleng;
    long sum = 0;
    int A[100];
    // name 의 글자 j 에 대하여 다음을 수행한다:
    leng = strlen(recname);
    for (j = 0; j < leng; j++) {
        u = recname[j];
        A[j] = u;
    }
    halfleng = leng / 2;
    for (j = 0; j < halfleng; j++)
        sum = sum + (A[j] * A[leng - 1 - j] * A[(leng - 1) / 2]);
    if (leng % 2 == 1)
        sum = sum + A[halfleng] * A[halfleng + 1] * A[(leng - 1) / 2];
    HA = sum % Tbl_Size; // HA is a home address given by the hash function.
    return HA;
}
```

name 을 구성하는 각 byte 를 unsigned char 로 보아서 접어서 매칭 되는 원소끼리 곱하여 그들의 총합을 구하고, 다음 이들의 총합을 table size Tbl_Size 로 나눈 나머지를 해싱의 결과 값으로 한다.

• 작업 수행 순서:

(1) 저장 단계:

(전체 레코드들의 삽입작업) "Companies_Korean.txt" 파일에서 이름을 하나씩 읽어 해시 테이블에 삽입한다. 이때, 이름 중간에 blank가 있을 수 있으므로 fgets를 이용하여 파일에서 이름을 읽는다. 이 이름을 레코드의 name 필드에 넣는다. (그리고 레코드의 monincome 필드도 랜덤 넘버로 채운다). 그리고 이 레코드를 해시테이블에 삽입한다.

(2) 명령문 실행 단계 (반복 루프):

명령들: i (삽입), r (검색), d(삭제), v(성능 측정), q(다중 삭제), c(연결 수 확인), e(종료).

[**각 명령의 설명**]

- i : 상호명을 하나 입력 받아서 이를 hash table 에 삽입한다.
  삽입 과정에서 접근한(내용을 조사한) 주소의 갯수 (probe 수)를 출력한다.
  동일한 키가 이미 존재하면 "삽입실패"를 출력한다.

- r: 상호명을 받아서 이를 탐색한다. 저장위치 및 탐색과정에서의 사용한 probe 수를 출력한다.
  (주: Hash table 에서 하나의 위치를 조사하는 행위를 하면 한 번의 probe 를 한 것이 된다.)

- d: 상호명을 받아서 이를 찾아서 삭제한다. 삭제 중에 발생한 record move 횟수를 출력한다.
  chain split 가 일어났으면 이 사실이 일어났음을 보고한다.

- v: 화일 Companies_Korean.txt 의 처음부터 하나씩 이름을 읽어서 탐색을 수행한다.
  탐색과정에서 발생한 probe의 수 중 최대값을 구한다.

- q: 다중 삭제 명령이다. 수를 입력받아서 이 수 만큼을 줄을 주어진 이름 파일에서 읽어서 이 이름들을 삭제한다. 성공한 삭제 수와 레코드 이동이 필요한 삭제의 수를 출력해 준다.

- c: link 의 수를 확인하는 명령이다. 상호명을 하나 입력 받아서 해당 상호명과 연결되어 있는 레코드의 개수를 출력한다.

- 실험 예; Probe란 테이블에서 레코드를 읽는 작업 하나를 말한다.
  (주: r 과 d 명령은 각각 3회 이상씩 시도해 볼 것)

```
명령은? r 한국디젤
    Search succeeded. Loc_found=34987, n_probes=2
명령은? v
    maximum number of probes per search = 16
명령은? d 한국레저협회
    Deletion succeeded. Num_relocations = 1, num_chain_splits=0
명령은? r 한국레저협회
    No such record exists.
명령은? i 한국레저협회
    insertion succeeded. num_probe = 4
명령은? c 동양콘트렉트
    Number of links : 5
명령은? q 30000
    num_deletion_success=30000, num_relocated_deletions=12544
명령은? e
    실험을 종료합니다.
```

주1: 일부를 가린 guide 프로그램을 제공하니 참고해서 개발해도 좋음.

가린 부분:

```
/****************
to
be
filled
*****************/
```

주2: 제출방법: 프로그램 전체를 하나의 .c 파일에 담아서 이를 업로드 한다 (파일명 맨 뒤에 학번과 이름 넣을 것).

프로그래밍 개발에서 지켜야 할 내용 :

- 빈자리가 필요할 때 이 위치를 구하는 방법은 가장 주소 값이 큰 빈 자리를 이용하도록 한다.

- 데이터 파일 "Companies_Korean.txt"는 solution 파일과 동일한 디렉토리에 있다고 가정하고 프로그램을 개발하여야 한다.

- 이 과제에서는 다음과 같은 작업을 하는 함수들을 작성하고 이를 호출하여 시스템을 개발하도록 한다.

1. **insert_rec** : 레코드 하나를 받아서 이를 해시 테이블에 저장한다.
   int insert_rec (type_record a_rec) {.....}
   반환값은 number of probes 이다. (-1이면 삽입 실패)

2. **retrieve_rec** : key(즉 스트링)을 받아서 이를 키로 하는 레코드를 찾는다.
   int retrieve_rec (char *key, int *probe) {.....}
   반환값: key 를 가지는 레코드의 table 내의 index (위치). 못찾으면 -1.
   probe: 찾는 과정에서 일어난 총 probe 수.

3. **delete_rec** : key를 받아서 이를 가지는 레코드를 해시 테이블에서 제거한다.
   int delete_rec (char *key, int *split) {.....}
   반환값: 위치 이동이 일어난 레코드 수.
   split: chain split 의 발생 여부 (1 or 0)

4. **count_max_probe** : 탐색의 최대 프로브 수

   int count_max_probe(const char *filename) {.....}
   반환값 : 탐색 최대 프로브 수

5. **check_num_links** : 해당 키값과 연관된 레코드의 수 (서로 연결되어있는 레코드의 수)를 계산한다.
   int check_num_links(char key[]) {.....}
   반환값: 입력받은 key값과 연결되어 있는 레코드의 수

- 이 프로그램에서 line의 타입은 int이다. Line의 값이 0~(Tbl_size – 1) 중의 하나라면 다른 위치를 정상적으로 가리키고 있다. 우리는 NULL 포인터에 대응하는 값으로 -1을 이용한다. 즉 link에 -1 이 들어 있다면 NULL이 들어 있는 것과 같이 취급한다. (즉 다른 위치를 가리키지 않는다는 의미이다).

- 해시 테이블의 각 위치가 비어 있는 위치 (빈 자리)임을 표시하는 방법
  빈 위치는 name 필드의 name[0]에 '₩0' 를 넣어 놓아서 표시한다.
  예를 들면, 위치 pos가 비어 있는지는 Hashtable[pos].name[0] 을 체크하면 된다. 여기에 null 문자('₩0')가 들어 있으면 이 위치 Hashtable[pos] 는 빈 자리이다.
  따라서 맨 처음에는 Hashtable 의 모든 원소의 name[0] 에 '₩0'를 넣어 놓아야 한다.