자료구조 실습과제 19

- 솔루션 및 프로젝트 명칭 : Proj_19_이름이니셜

- 제출방법 : 아래 문제를 해결하기 위한 프로그램을 구현한 후 컴파일 및 실행한 후, 오류가 없으면 메뉴에서 **솔루션 정리를 수행한 후 윈도우 탐색기에서 솔루션 폴더를 찾아 압축하여** E-class에 올림

문제 1) 해싱 기법을 이용한 데이터 저장 및 충돌 해결 방안 구현

- 저장하고자 하는 데이터의 키를 해시 함수를 적용하여 저장되는 위치를 계산하고, 그 위치에 데이터 레코드를 저장하는 방법이 해싱이다. 해싱에서는 서로 다른 키에 해시 함수를 적용하였을 때 동일한 저장위치가 나오는 경우 충돌이 발생하며, 이를 해결하기 위한 방법으로 개방주소법과 체인기법이 있다. 이번 실습과제는 해시를 통해 데이터를 저장하고, 충돌이 발생한 경우 해결하는 두가지 기법을 구현하는 것이다.

- 개방형주소법에서는 이중해싱을 이용하여 충돌이 발생한 경우 다음 저장 위치를 찾는다. 체인기법에서는 충돌이 발생하면 현재 연결리스트의 마지막에 데이터를 추가한다. 해시 테이블에 동일한 키 값이 이미 들어있다면 무시한다. 구현을 간단히 하기 위해서 데이터 레코드는 정수 값을 키로 가지는 키 값만 저장하기로 한다.

- 구현할 함수는 해시 테이블에 데이터를 저장하는 함수와 검색하는 함수이다. main() 함수의 예를 보고 필요한 함수를 구현하시오. 해시 함수는 간단하게 “키 % 테이블크기”로 정한다. 이중해싱에서는 2차 함수를 테이블크기보다 작은 가장 큰 소수 - (키 % 테이블크기보다 작은 가장 큰 소수)로 정하고, 충돌이 발생하면 이 값을 계속 더하여 다음 저장 위치를 찾는다.

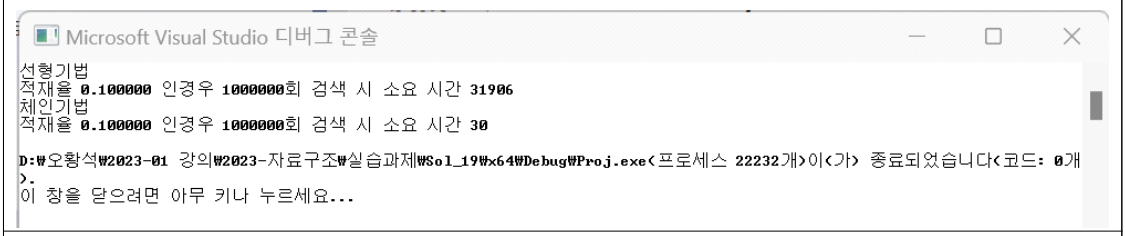
실행 예와 같이 적재률을 높여 가면서 두 기법의 검색 시간을 비교하시오.

```
// 해싱 테이블의 내용을 출력
#define TABLE_SIZE      10000   // 가능하면 해싱 테이블의 크기를 소수로 설정
typedef struct {
        int key;
} element;

struct list
{
        element item;
        struct list* link;
};

// 제산 함수를 사용한 해싱 함수
int hash_function(int key)
```

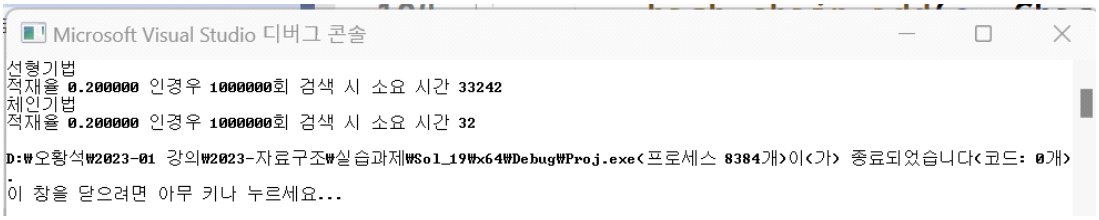
```
{
	return key % TABLE_SIZE;
}

#define SIZE TABLE_SIZE/2
#define SEARCH_COUNT 1000000
struct list** Chash_table;
element *Lhash_table;

// 해싱 테이블을 사용한 예제
int main(void)
{
	int s_time, e_time;
	int h_prime;
	element e;

	Lhash_table = (element*)malloc(sizeof(element) * TABLE_SIZE);
	Chash_table = (struct list**)malloc(sizeof(struct list*) * TABLE_SIZE);
	init_Lhash_table(Lhash_table, TABLE_SIZE);
	init_Chash_table(Chash_table, TABLE_SIZE);

	srand(100);
	h_prime = get_prime_number(TABLE_SIZE); //이중 해시법을 적용하기 위한 버킷 사이즈보다 작은 가장 큰 소수 구하기

	for (int i = 0; i < SIZE; i++) {
		e.key = rand();
		hash_linear_add(e, Lhash_table, h_prime);
		hash_chain_add(e, Chash_table);
	}

	srand(200);
	s_time = clock();
	for (int i = 0; i < SEARCH_COUNT; i++) {
		e.key = rand();
		hash_linear_search(e, Lhash_table, h_prime);
	}
	e_time = clock();

	printf("선형기법\n");
	printf("적재율 %f 인경우 %d회 검색 시 소요 시간 %d \n", (float)SIZE/TABLE_SIZE,
		SEARCH_COUNT, e_time - s_time);

	srand(200);
	s_time = clock();
	for (int i = 0; i < SEARCH_COUNT; i++) {
		e.key = rand();
		hash_chain_search(e, Chash_table);
	}
	e_time = clock();

	printf("체인기법\n");
	printf("적재율 %f 인경우 %d회 검색 시 소요 시간 %d \n ", (float)SIZE / TABLE_SIZE,
		SEARCH_COUNT, e_time - s_time);
```

```
        return 0;
}
```

실행 예 :



실행 예 :



실행 예 :