Sem	Nombre archivo	Descripción	Comentario
6	table-lookup.h	Crear un directorio table-lookup. Todos los archivos de este ejercicio deben residir ahí. Copiar del libro las definiciones de struct nlist y HASHSIZE, luego agregar las declaraciones (solo las declaraciones, no el código que es la 2da parte) de *hash*, *lookup* e *install* del ejercicio propuesto en el capítulo 6.6: inserción de nombres y textos de reemplazo (similar a lo que debe hacer el compilador C con las sentencias #define) en una tabla de hash. Guardar esto en el archivo table-lookup.h	Ejercicio domiciliario, 1ra parte.
6	table-lookup-func.c	Copiar del libro el código de las funciones *hash*, *lookup* e *install* del ejercicio propuesto en el capítulo 6.6. Escribir estas funciones en el archivo *table-lookup-func.c*.	Ejercicio domiciliario, 2da parte.
6	table-lookup-func.c	Agregar al archivo *table-lookup-func.c*, la función *void undef(char *name)* que elimina el nombre *name* y su definición de la tabla de hash mantenida con con *install* y *lookup*.	Ejercicio domiciliario, 3a parte.
6	table-lookup.c getword.c	Armar un programa principal (función main), cuyo cuerpo principal está listado más abajo. Recuerde declarar las variables con los tipos adecuados. Copiar el cuerpo de la función *getword* que hemos usado en ejercicios anteriores, en un archivo aparte *getword.c*.	Ejercicio domiciliario, 4ta parte.
6	Makefile	Armar un Makefile para compilar y generar el ejecutable table-lookup.	Ejercicio domiciliario, 5ta parte.

función main de table-lookup.c – loop de lectura e inserción en la tabla

```
for (nword=1; getword(word, MAXWORD) != EOF; nword++) {
    if (isalpha(word[0])) {
        snprintf(str_nword, MAXWORD, "%d", nword);
        if (install(word, str_nword) == NULL) {
            fprintf(stderr, "Error en install\n");
        }
    }
}
```

función main de table-lookup.c – loop que imprime y borra los elementos de la tabla

```
for (i=0; i<HASHSIZE; i++) {
    np = hashtab[i];
    while (np != NULL) {
        printf("%s: %s\n", np->name, np->defn);
        aux = np->next;     // guardo np->next antes de hacer undef
        undef(np->name);
        np = aux;
    }
}
```