```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//****************          FUNCIONES PARA LA PILA
// Definicion de la estructura Nodo
struct NodoP {
    int dato;
    struct NodoP *siguiente;
};

// Funcion para agregar un nuevo elemento a la pila
void push(struct NodoP **top, int dato) {
    struct NodoP *nuevoNodo = (struct NodoP *) malloc(sizeof(struct NodoP));
    nuevoNodo->dato = dato;
    nuevoNodo->siguiente = *top;
    *top = nuevoNodo;
}

// Funcion para quitar el ultimo elemento de la pila
int pop(struct NodoP **top) {
    int dato = -1;
    if (*top == NULL) {
        printf("La pila esta vacia\n");
    } else {
        struct NodoP *temp = *top;
        *top = (*top)->siguiente;
        dato = temp->dato;
        free(temp);
    }
    return dato;
}

// Funcion para mostrar los elementos de la pila
void mostrarPila(struct NodoP *top) {
    printf("Pila: ");
    while (top != NULL) {
        printf("%d ", top->dato);
        top = top->siguiente;
    }
    printf("\n");
}

//**************          FUNCIONES PARA LA COLA
#define DEFAULT_QUEUE_SIZE 10

struct Queue {
    int *items;
    int front;
    int rear;
    int size;
};

typedef struct Queue queue_t;

// Funcion para agregar un elemento a la cola
void enqueue(queue_t *q) {
    int item;
    printf("Ingrese el elemento que desea agregar: ");
    scanf("%d", &item);
```

```
60
61      if (q->rear == q->size - 1) {
62          printf("La cola esta llena\n");
63      } else {
64          if (q->front == -1) {
65              q->front = 0;
66          }
67          q->rear++;
68          q->items[q->rear] = item;
69          printf("Elemento agregado a la cola: %d\n", item);
70      }
71  }
72
73  // Funcion para quitar un elemento de la cola
74  int dequeue(queue_t *q) {
75      int item;
76      if (q->front == -1 || q->front > q->rear) {
77          printf("La cola esta vacia\n");
78          item = -1;
79      } else {
80          item = q->items[q->front];
81          q->front++;
82          printf("Elemento removido de la cola: %d\n", item);
83      }
84      return item;
85  }
86
87  // Funci?n para mostrar los elementos de la cola
88  void display(queue_t *q) {
89      if (q->rear == -1) {
90          printf("La cola esta vacia\n");
91      } else {
92          int i;
93          printf("Elementos en la cola: ");
94          for (i = q->front; i <= q->rear; i++) {
95              printf("%d ", q->items[i]);
96          }
97          printf("\n");
98      }
99  }
100
101 // Funcion para cambiar el tamano de la cola
102 void resize(queue_t *q) {
103     int new_size;
104     printf("Ingrese el nuevo tamano de la cola: ");
105     scanf("%d", &new_size);
106
107     if (new_size < q->rear - q->front + 1) {
108         printf("No se puede reducir el tamano de la cola por debajo del numero
    actual de elementos\n");
109
110     } else {
111         q->items = (int *)realloc(q->items, new_size * sizeof(int));
112         q->size = new_size;
113         printf("Tamano de la cola actualizado a %d\n", new_size);
114     }
115 }
116
117 //*****************************       TABLAS HASH
118 #define MAX 50 // Tamano maximo de la tabla hash
```

```
119
120 // Definicion de la estructura nodo
121 typedef struct nodo {
122     char nombre[50]; // Nombre del elemento
123     int valor; // Valor del elemento
124     struct nodo *sig; // Puntero al siguiente nodo
125 } Nodo;
126
127 // Funcion hash
128 int funcion_hash(char *clave) {
129     int valor = 0;
130     for (int i = 0; i < strlen(clave); i++) {
131         valor += clave[i];
132     }
133     return valor % MAX;
134 }
135
136 // Funcion para crear un nodo
137 Nodo* crear_nodo(char *nombre, int valor) {
138     Nodo *nuevo_nodo = (Nodo*) malloc(sizeof(Nodo));
139     strcpy(nuevo_nodo->nombre, nombre);
140     nuevo_nodo->valor = valor;
141     nuevo_nodo->sig = NULL;
142     return nuevo_nodo;
143 }
144
145 void insertar_elemento(Nodo **tabla_hash, char *nombre, int valor) {
146     int indice = funcion_hash(nombre); // Obtenemos el indice
147     Nodo *nuevo_nodo = crear_nodo(nombre, valor); // Creamos el nuevo nodo
148     if (tabla_hash[indice] == NULL) { // Si la lista en esa posicion esta vacia,
    simplemente insertamos el nuevo nodo
149         tabla_hash[indice] = nuevo_nodo;
150     } else { // Si la lista en esa posicion no esta vacia, manejamos la colision
    mediante una lista enlazada
151         Nodo *nodo_actual = tabla_hash[indice];
152         while (nodo_actual->sig != NULL) {
153             nodo_actual = nodo_actual->sig; // Buscamos el final de la lista
154         }
155         nodo_actual->sig = nuevo_nodo; // Insertamos el nuevo nodo al final de la
    lista
156     }
157 }
158
159 // Funcion para buscar un elemento en la tabla hash
160 Nodo* buscar_elemento(Nodo **tabla_hash, char *nombre) {
161     int indice = funcion_hash(nombre); // Obtenemos el indice
162     Nodo *nodo_actual = tabla_hash[indice]; // Empezamos por el primer nodo de la
    lista en esa posicion
163     while (nodo_actual != NULL) {
164         if (strcmp(nodo_actual->nombre, nombre) == 0) { // Si encontramos el nodo
    con ese nombre, lo devolvemos
165             return nodo_actual;
166         }
167         nodo_actual = nodo_actual->sig; // Pasamos al siguiente nodo
168     }
169     return NULL; // Si no encontramos el nodo con ese nombre, devolvemos NULL
170 }
171
172 // Funci?n para mostrar la tabla hash
173 void mostrar_tabla(Nodo **tabla_hash) {
```

```
174     for (int i = 0; i < MAX; i++) {
175         printf("Posicion %d: ", i);
176         Nodo *nodo_actual = tabla_hash[i]; // Empezamos por el primer nodo de la
    lista en esa posicion
177         while (nodo_actual != NULL) {
178             printf("(%s, %d) ", nodo_actual->nombre, nodo_actual->valor);
179             nodo_actual = nodo_actual->sig; // Pasamos al siguiente nodo
180         }
181         printf("\n");
182     }
183 }
184 
185 //*****************************          FUNCION PRINCIPAL
186 int main() {
187   //Menu principal
188   MenuPrincipal:
189   system("cls");
190   int num;
191   int cant;
192   int select;
193   printf("Selecciona una opcion:\n1 = PILAS\n2 = COLAS\n3 = TABLAS HASH\n4 =
    Salir\n");
194   scanf("%d",&select);
195   switch(select){
196     case 1:
197       {
198         system("cls");
199         struct NodoP *top = NULL;
200         printf("cuantos datos quiere agregar a la pila?\n");
201         scanf("%d",&cant);
202         //Loop para agregar los elementos a la pila
203         for (int i=0;i<cant;i++){
204           printf("\nIngrese un numero a la pila\n");
205           scanf("%d",&num);
206           push(&top,num);
207         }
208         MenuPilas:
209         system("cls");
210         printf("Elige una opcion:\n1 = Mostrar Pila\n2 = Eliminar elemento\n3 =
    Agregar Elemento\n4 = Menu Principal\n");
211         scanf("%d",&select);
212         switch(select){
213 
214           case 1:
215             { // Mostrar los elementos de la pila
216               system ("cls");
217                 mostrarPila(top);
218                 system("pause");
219                 goto MenuPilas;
220               }
221 
222           case 2:
223             { // Eliminar un elemento de la pila
224               system("cls");
225                 int dato = pop(&top);
226                 printf("Elemento eliminado: %d\n", dato);
227                 system("pause");
228                 goto MenuPilas;
229               }
230 
```

```
231           case 3:
232             { //agregar elemento
233               printf("Ingresa el elemento: \n");
234               scanf("%d", &num);
235               push(&top,num);
236               goto MenuPilas;
237             }
238 
239           case 4:
240             { //Volver al menu principal
241               goto MenuPrincipal;
242             }
243 
244           default:
245             {
246               goto MenuPilas;
247             }
248         }
249       }
250 
251     case 2:
252       {
253         system("cls");
254         queue_t q;
255           q.front = -1;
256           q.rear = -1;
257           q.size = DEFAULT_QUEUE_SIZE;
258           q.items = (int *)malloc(q.size * sizeof(int));
259         MenuCola:
260         system("cls");
261           printf("Elige una opcion:\n1 = Agregar Elemento\n2 = Quitar Elemento\n3 =
    Mostrar Cola\n4 = Cambiar Tama?o\n5 = Menu Principal\n");
262           scanf("%d", &select);
263           switch(select){
264             case 1:
265               { //Agregar elemento
266                 system("cls");
267                 enqueue(&q);
268               system("pause");
269               goto MenuCola;
270             }
271 
272           case 2:
273             { //Quitar elemento
274               system("cls");
275               dequeue(&q);
276               system("pause");
277               goto MenuCola;
278             }
279 
280           case 3:
281             { //Mostrar Cola
282               system("cls");
283               display(&q);
284               system("pause");
285               goto MenuCola;
286             }
287 
288           case 4:
289             { //Cambiar tamano de cola
```

```
290               system("cls");
291               resize(&q);
292               system("pause");
293               goto MenuCola;
294             }
295
296           case 5:
297             { //Ir al Menu Principal
298               goto MenuPrincipal;
299             }
300
301           default:
302             goto MenuCola;
303         }
304       }
305     case 3:
306       {
307         Nodo *tabla_hash[MAX]; // Inicializamos la tabla hash
308           for (int i = 0; i < MAX; i++) {
309               tabla_hash[i] = NULL; // Inicializamos cada lista como vacia
310           }
311           char nombre[50];
312           int valor;
313           MenuTablaHash:
314           system("cls");
315         printf("Elige una opcion:\n1 = Agregar elementos a la tabla\n2 = Buscar
elementos en la tabla\n3 = Mostrar tabla\n4 = Menu Principal\n");
316           scanf("%d",&select);
317           switch (select){
318             case 1:
319               { // Pedimos al usuario que ingrese los datos
320                 system("cls");
321                 printf("Ingrese el nombre y valor del elemento (para salir, ingrese
'salir' como nombre): ");
322                 scanf("%s", nombre);
323                 while (strcmp(nombre, "salir") != 0) { // Mientras el usuario no
ingrese "salir"
324                   printf("\nIngrese el valor del elemento: ");
325                     scanf("%d", &valor);
326                     insertar_elemento(tabla_hash, nombre, valor); // Insertamos el
elemento en la tabla hash
327                     printf("Ingrese el nombre y valor del elemento (para salir,
ingrese 'salir' como nombre): ");
328                     scanf("%s", nombre);
329                 }
330                 system("pause");
331                 goto MenuTablaHash;
332             }
333
334           case 2:
335             { // Buscamos un elemento en la tabla hash
336               system("cls");
337                 printf("Ingrese el nombre del elemento a buscar: ");
338                 scanf("%s", nombre);
339                 Nodo *nodo_buscado = buscar_elemento(tabla_hash, nombre);
340                 if (nodo_buscado != NULL) {
341                     printf("El valor del elemento %s es %d.\n", nodo_buscado-
>nombre, nodo_buscado->valor);
342                 } else {
343                     printf("El elemento no se encontro en la tabla hash.\n");
```

```
344                 }
345               system("pause");
346               goto MenuTablaHash;
347             }
348
349           case 3:
350             { //Mostramos la tabla
351               system("cls");
352               mostrar_tabla(tabla_hash);
353               system("pause");
354               goto MenuTablaHash;
355             }
356
357           case 4:
358             {
359               goto MenuPrincipal;
360             }
361
362           default:
363             {
364               goto MenuTablaHash;
365             }
366         }
367       }
368
369     case 4:
370       {
371         break;
372       }
373
374     default:
375         {
376           goto MenuPrincipal;
377       }
378   }
379     return 0;
380 }
```