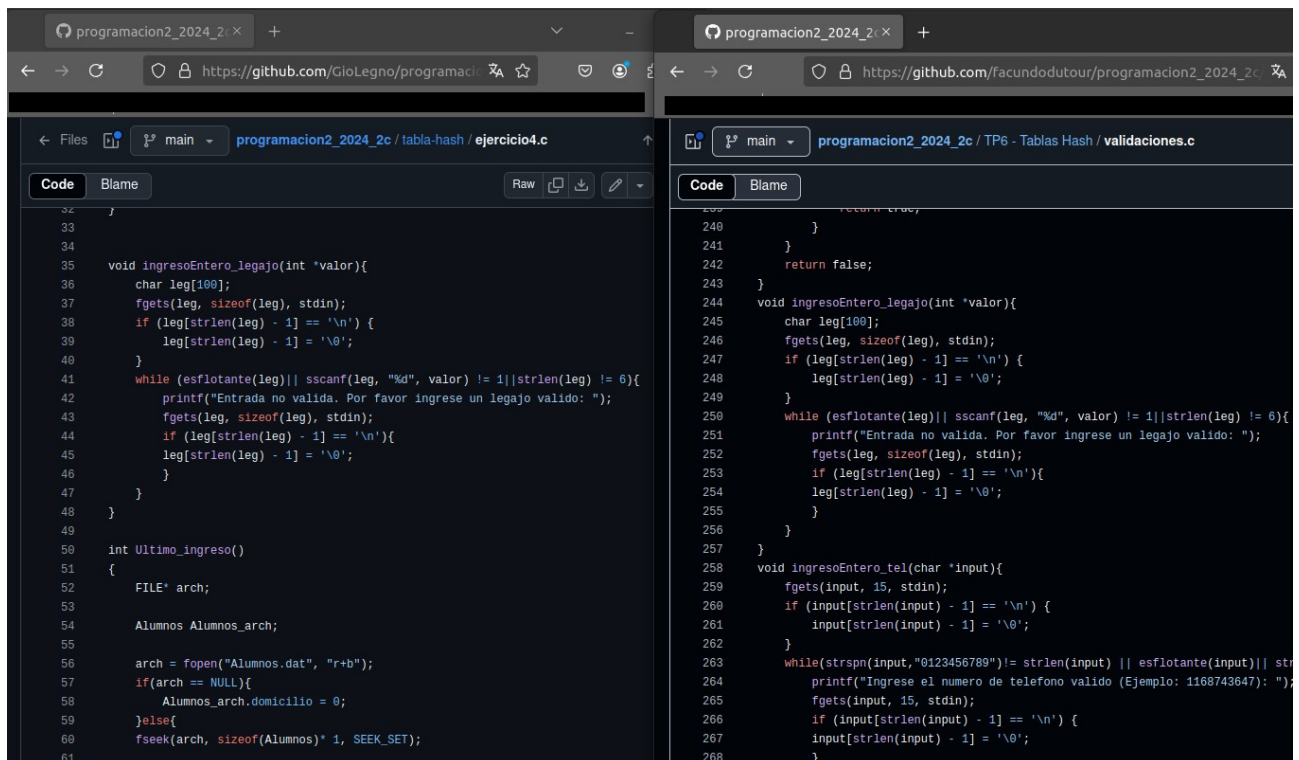


RESULTADO DE LA CORRECCIÓN: **DESAPROBADO**

## OBSERVACIONES

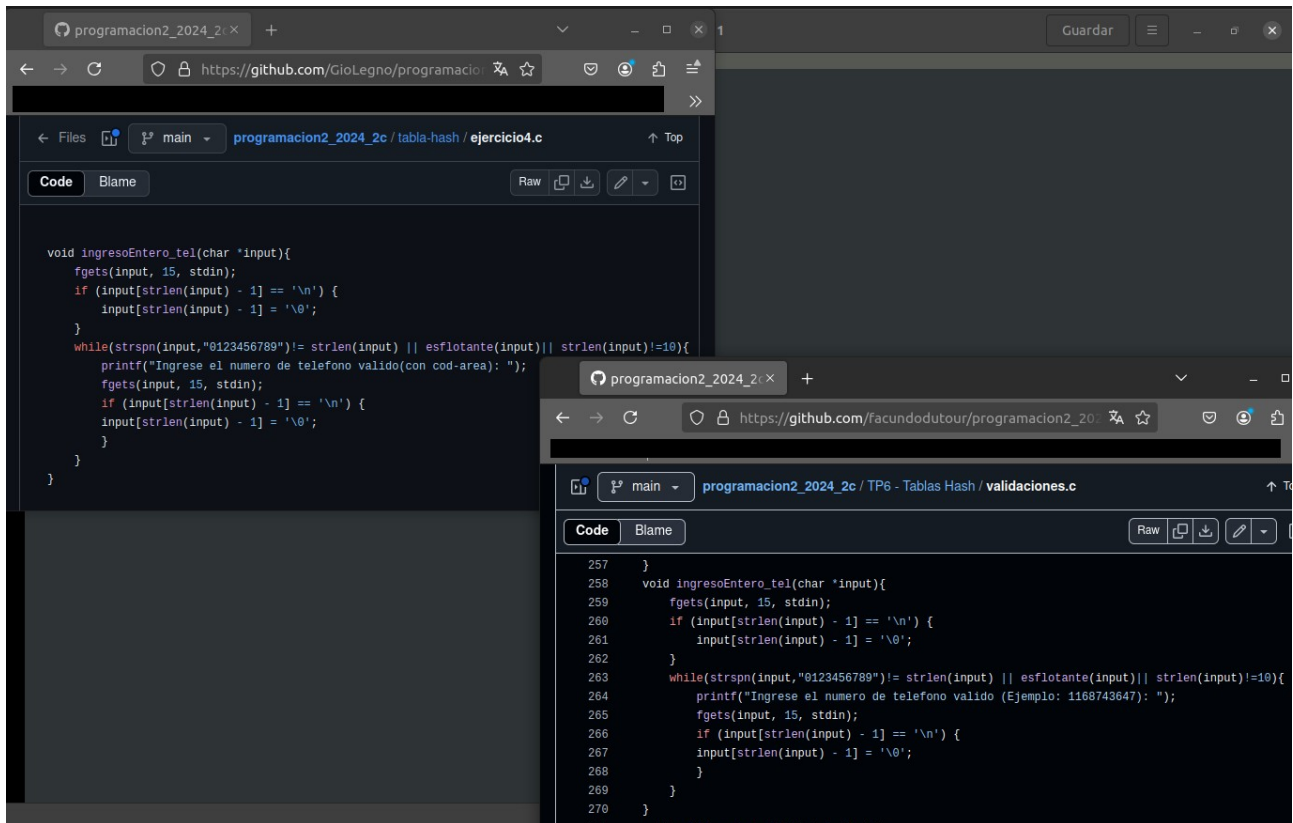
En el ejercicio 4 validan innecesariamente el teléfono, hay que mirar el código para entender el formato permitido. Piden la posición de un alumno para modificarlo en lugar del legajo. En la opción de menú para mostrar los datos del alumno, pide la posición y en teoría lo borra, pero en la tabla sigue estando. En el ejercicio 5 no controla rango de claves, por cada experimento pide las cantidades de claves y rangos. En el ejercicio 6 no controla las fechas, permite cargar el mismo DNI en la misma fecha.

LA RESOLUCIÓN DEL EJERCICIO 4 ES IGUAL A LA PRESENTADA POR EL GRUPO 10



```
32 }
33
34
35 void ingresoEntero_legajo(int *valor){
36     char leg[100];
37     fgets(leg, sizeof(leg), stdin);
38     if (leg[strlen(leg) - 1] == '\n') {
39         leg[strlen(leg) - 1] = '\0';
40     }
41     while (esflotante(leg) || sscanf(leg, "%d", valor) != 1 || strlen(leg) != 6){
42         printf("Entrada no valida. Por favor ingrese un legajo valido: ");
43         fgets(leg, sizeof(leg), stdin);
44         if (leg[strlen(leg) - 1] == '\n'){
45             leg[strlen(leg) - 1] = '\0';
46         }
47     }
48 }
49
50 int Ultimo_ingreso()
51 {
52     FILE* arch;
53
54     Alumnos Alumnos_arch;
55
56     arch = fopen("Alumnos.dat", "r+b");
57     if(arch == NULL){
58         Alumnos_arch.domicilio = 0;
59     }else{
60         fseek(arch, sizeof(Alumnos)* 1, SEEK_SET);
61
240     }
241 }
242 return false;
243 }
244 void ingresoEntero_legajo(int *valor){
245     char leg[100];
246     fgets(leg, sizeof(leg), stdin);
247     if (leg[strlen(leg) - 1] == '\n') {
248         leg[strlen(leg) - 1] = '\0';
249     }
250     while (esflotante(leg) || sscanf(leg, "%d", valor) != 1 || strlen(leg) != 6){
251         printf("Entrada no valida. Por favor ingrese un legajo valido: ");
252         fgets(leg, sizeof(leg), stdin);
253         if (leg[strlen(leg) - 1] == '\n'){
254             leg[strlen(leg) - 1] = '\0';
255         }
256     }
257 }
258 void ingresoEntero_tel(char *input){
259     fgets(input, 15, stdin);
260     if (input[strlen(input) - 1] == '\n') {
261         input[strlen(input) - 1] = '\0';
262     }
263     while (strspn(input, "0123456789") != strlen(input) || esflotante(input) || str
264         printf("Ingrese el numero de telefono valido (Ejemplo: 1168743647): ");
265     fgets(input, 15, stdin);
266     if (input[strlen(input) - 1] == '\n') {
267         input[strlen(input) - 1] = '\0';
268     }
```

## GRUPO 5 – Correcciones Trabajo Práctico: TABLAS HASH

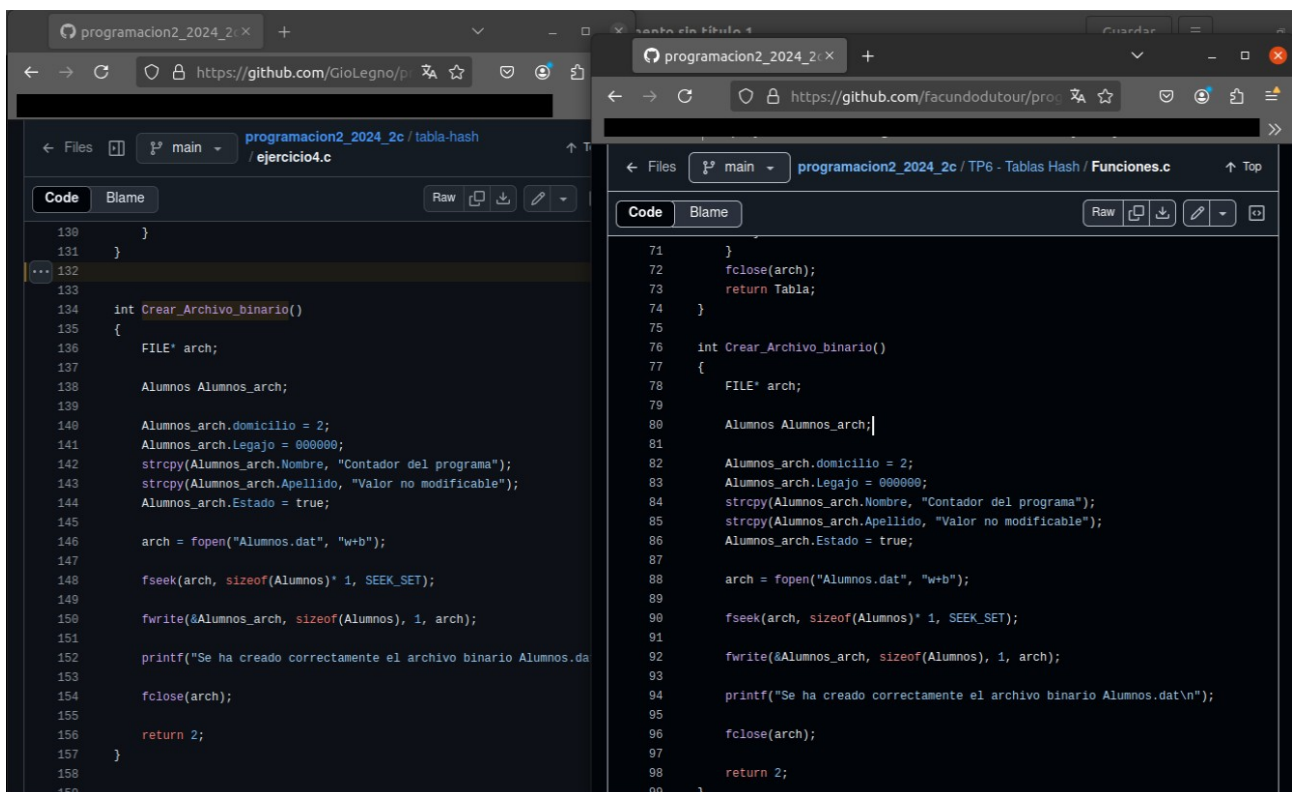


The top-left window shows the file `programacion2_2024_2c / tabla-hash / ejercicio4.c`. The code defines a function `ingresoEntero_tel` that reads a phone number from standard input and validates it. The validation logic checks for a specific area code and ensures the number is 10 digits long.

```
void ingresoEntero_tel(char *input){
    fgets(input, 15, stdin);
    if (input[strlen(input) - 1] == '\n') {
        input[strlen(input) - 1] = '\0';
    }
    while(strspn(input,"0123456789")!= strlen(input) || esflotante(input) || strlen(input)!=10){
        printf("Ingrese el numero de telefono valido(con cod-area): ");
        fgets(input, 15, stdin);
        if (input[strlen(input) - 1] == '\n') {
            input[strlen(input) - 1] = '\0';
        }
    }
}
```

The top-right window shows the file `programacion2_2024_2c / TP6 - Tablas Hash / validaciones.c`. It contains a similar validation function, also named `ingresoEntero_tel`, with slightly different validation logic.

```
257 }
258 void ingresoEntero_tel(char *input){
259     fgets(input, 15, stdin);
260     if (input[strlen(input) - 1] == '\n') {
261         input[strlen(input) - 1] = '\0';
262     }
263     while(strspn(input,"0123456789")!= strlen(input) || esflotante(input) || strlen(input)!=10){
264         printf("Ingrese el numero de telefono valido (Ejemplo: 1168743647): ");
265         fgets(input, 15, stdin);
266         if (input[strlen(input) - 1] == '\n') {
267             input[strlen(input) - 1] = '\0';
268         }
269     }
270 }
```



The bottom-left window shows the file `programacion2_2024_2c / tabla-hash / ejercicio4.c`. It contains a function `Crear_Archivo_binario` that creates a binary file named `Alumnos.dat` and writes an `Alumnos_arch` structure to it.

```
139 }
140 }
141 }
142 ...
143 int Crear_Archivo_binario()
144 {
145     FILE* arch;
146     Alumnos Alumnos_arch;
147     Alumnos_arch.domicilio = 2;
148     Alumnos_arch.Legajo = 000000;
149     strcpy(Alumnos_arch.Nombre, "Contador del programa");
150     strcpy(Alumnos_arch.Apellido, "Valor no modificable");
151     Alumnos_arch.Estado = true;
152     arch = fopen("Alumnos.dat", "w+b");
153     fseek(arch, sizeof(Alumnos)* 1, SEEK_SET);
154     fwrite(&Alumnos_arch, sizeof(Alumnos), 1, arch);
155     printf("Se ha creado correctamente el archivo binario Alumnos.dat\n");
156     fclose(arch);
157     return 2;
158 }
```

The bottom-right window shows the file `programacion2_2024_2c / TP6 - Tablas Hash / Funciones.c`. It contains a function `Crear_Archivo_binario` that is identical to the one in the previous window.

```
71 }
72 fclose(arch);
73 return Tabla;
74 }
75
76 int Crear_Archivo_binario()
77 {
78     FILE* arch;
79     Alumnos Alumnos_arch;
80     Alumnos_arch.domicilio = 2;
81     Alumnos_arch.Legajo = 000000;
82     strcpy(Alumnos_arch.Nombre, "Contador del programa");
83     strcpy(Alumnos_arch.Apellido, "Valor no modificable");
84     Alumnos_arch.Estado = true;
85     arch = fopen("Alumnos.dat", "w+b");
86     fseek(arch, sizeof(Alumnos)* 1, SEEK_SET);
87     fwrite(&Alumnos_arch, sizeof(Alumnos), 1, arch);
88     printf("Se ha creado correctamente el archivo binario Alumnos.dat\n");
89     fclose(arch);
90     return 2;
91 }
```

## GRUPO 5 – Correcciones Trabajo Práctico: TABLAS HASH

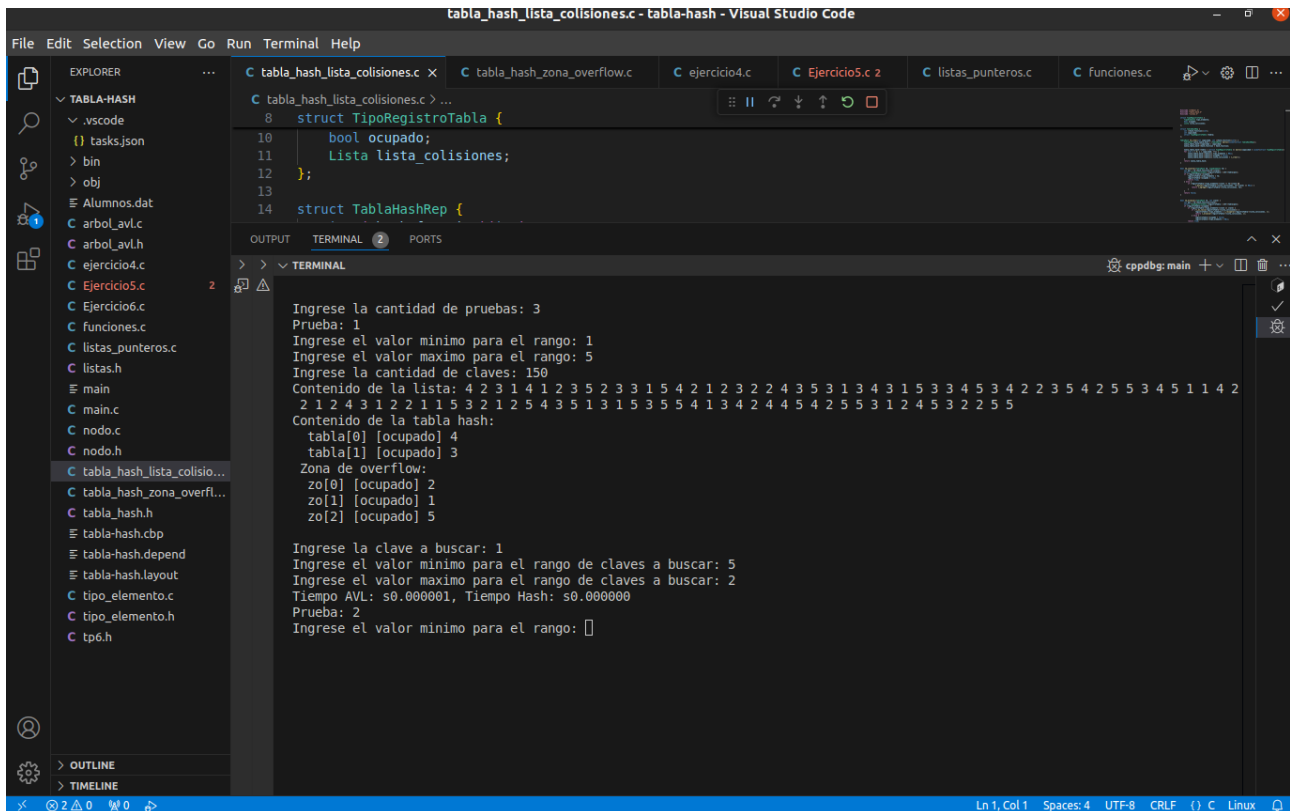
```
159
160 void Altas(int* Lugar_de_carga)
161 {
162     FILE* arch;
163
164     int Pos1 = *Lugar_de_carga;
165
166     Alumnos Alumnos_arch;
167
168     arch = fopen("Alumnos.dat", "r+b");
169     int flag = 1;
170
171     if (arch == NULL)
172     {
173         printf("\nNo existe el archivo\n");
174         flag = 0;
175         return;
176     }
177     int cargas;
178     printf("\nCuántas personas quiere cargar: ");
179     validacion_cargas(&cargas);
180     int i=0;
181     while (i<cargas)
182     {
183         printf("\nCargando alumno\n\n");
184
185         int legajo = 0;
186         printf("Ingrese legajo del alumno: ");
```

```
134 void Altas(int* Lugar_de_carga)
135 {
136     FILE* arch;
137
138     int Pos1 = *Lugar_de_carga;
139
140     Alumnos Alumnos_arch;
141
142     arch = fopen("Alumnos.dat", "r+b");
143     int flag = 1;
144
145     if (arch == NULL)
146     {
147         printf("\nNo existe el archivo\n");
148         flag = 0;
149         return;
150     }
151     int cargas;
152     printf("\nCuántas personas en total quisiera cargar?: ");
153     validacion_cargas(&cargas);
154     int i=0;
155     while (i<cargas)
156     {
157         printf("\nCargando alumno\n\n");
158
159         int legajo = 0;
160         printf("Ingrese legajo del alumno (no menos de 6 dígitos): ");
```

```
26 for (int i = 0; input2[i] != '\0' && input2[i] != '\n'; i++) {
27     if (!isalnum(input2[i])) {
28         valido = false;
29         printf("Entrada no valida. Solo se permiten numeros y le
30         break;
31     }
32 }
33 }
34 }
35
36 void validacion_cargas(int *valor){
37     char flag[100];
38     fgets(flag, sizeof(flag), stdin);
39     if (flag[strlen(flag) - 1] == '\n') {
40         flag[strlen(flag) - 1] = '\0';
41     }
42     while (esflotante(flag) || sscanf(flag, "%d", valor) != 1 || *valor<0){
43         printf("Entrada no valida. Por favor ingrese una opcion valida:
44         fgets(flag, sizeof(flag), stdin);
45     }
46 }
47
48 void validacion_opcion(int *valor)
49 {
50     char flag[100];
51     printf("Ingrese opcion: ");
52     fgets(flag, sizeof(flag), stdin);
53     if (flag[strlen(flag) - 1] == '\n') {
```

```
112 if (flag[strlen(flag) - 1] == '\n') {
113     flag[strlen(flag) - 1] = '\0';
114 }
115 while (esflotante(flag) || sscanf(flag, "%d", valor) != 1 || *valor<0 || *valor>2){
116     printf("Entrada no valida. Por favor ingrese una opcion valida: ");
117     fgets(flag, sizeof(flag), stdin);
118 }
119 }
120 void validacion_cargas(int *valor)
121 {
122     char flag[100];
123     fgets(flag, sizeof(flag), stdin);
124     if (flag[strlen(flag) - 1] == '\n') {
125         flag[strlen(flag) - 1] = '\0';
126 }
127 while (esflotante(flag) || sscanf(flag, "%d", valor) != 1 || *valor<0){
128     printf("Entrada no valida. Por favor ingrese una opcion valida: ");
129     fgets(flag, sizeof(flag), stdin);
130 }
131 }
132 bool es_max(int *min, int *max){
133     bool maxim=true;
134     if(*max<*min){
135         maxim=false;
136     }
137     return maxim;
138 }
```

## GRUPO 5 – Correcciones Trabajo Práctico: TABLAS HASH

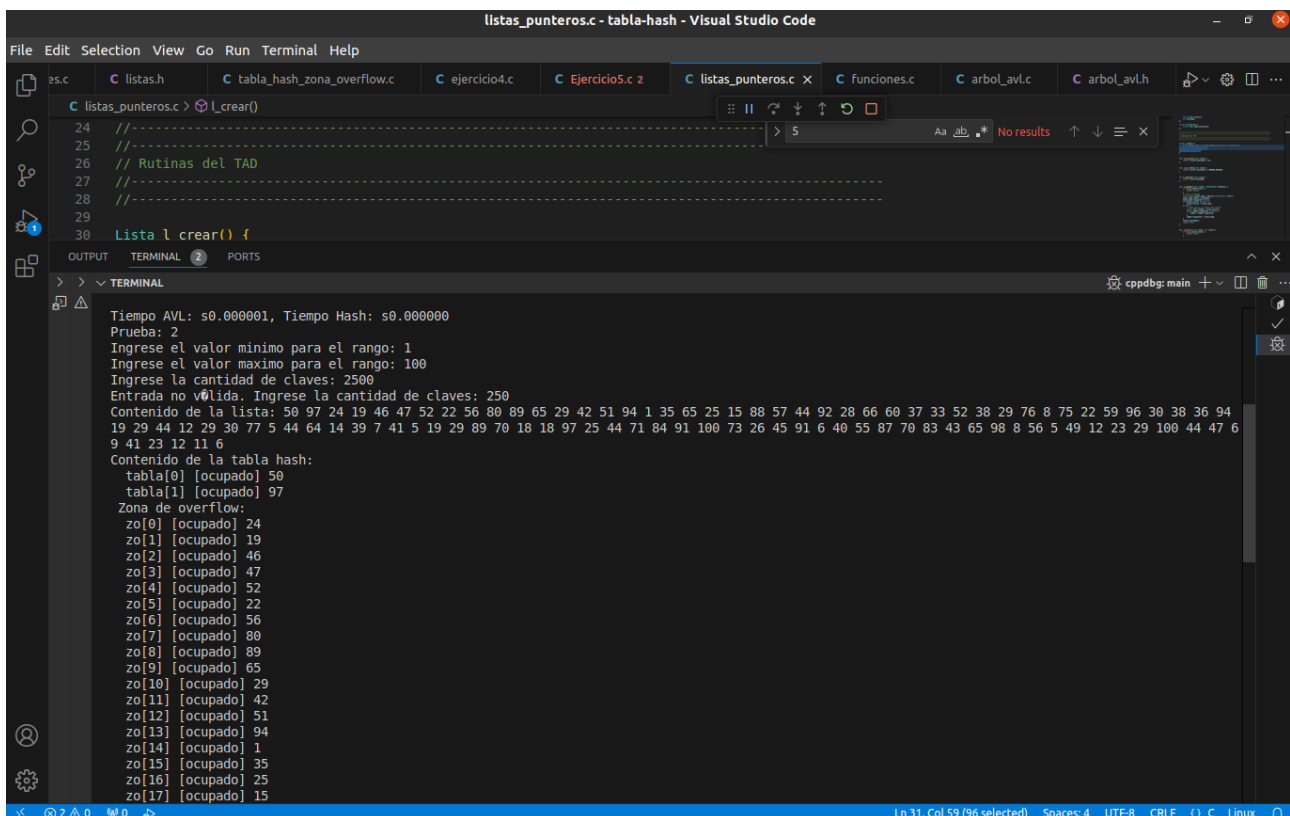


```
tabla_hash_lista_colisiones.c - tabla-hash - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  TABLA-HASH
  .vscode
  {} tasks.json
  > bin
  > obj
  Alumnos.dat
  arbol_avl.c
  arbol_avl.h
  ejercicio4.c
  ejercicio5.c
  Ejercicio6.c
  funciones.c
  listas_punteros.c
  listas.h
  main
  main.c
  nodo.c
  nodo.h
  tabla_hash_lista_colisio...
  tabla_hash_zona_overflow...
  tabla_hash.h
  tabla-hash.cbp
  tabla-hash.depend
  tabla-hash.layout
  tipo_elemento.c
  tipo_elemento.h
  tp6.h
  OUTLINE
  TIMELINE

C tabla_hash_lista_colisiones.c > ...
8 struct TipoRegistroTabla {
10     bool ocupado;
11     Lista lista_colisiones;
12 };
13
14 struct TablaHashRep {
15     ...
16 };

OUTPUT TERMINAL PORTS
> > > TERMINAL
Ingrese la cantidad de pruebas: 3
Prueba: 1
Ingrese el valor minimo para el rango: 1
Ingrese el valor maximo para el rango: 5
Ingrese la cantidad de claves: 150
Contenido de la lista: 4 2 3 1 4 1 2 3 5 2 3 3 1 5 4 2 1 2 3 2 2 4 3 5 3 1 3 4 3 1 5 3 3 4 5 3 4 2 2 3 5 4 2 5 5 3 4 5 1 1 4 2
2 1 2 4 3 1 2 2 1 1 5 3 2 1 2 5 4 3 5 1 3 1 5 3 5 5 4 1 3 4 2 4 4 5 4 2 5 5 3 1 2 4 5 3 2 2 5 5
Contenido de la tabla hash:
tabla[0] [ocupado] 4
tabla[1] [ocupado] 3
Zona de overflow:
zo[0] [ocupado] 2
zo[1] [ocupado] 1
zo[2] [ocupado] 5

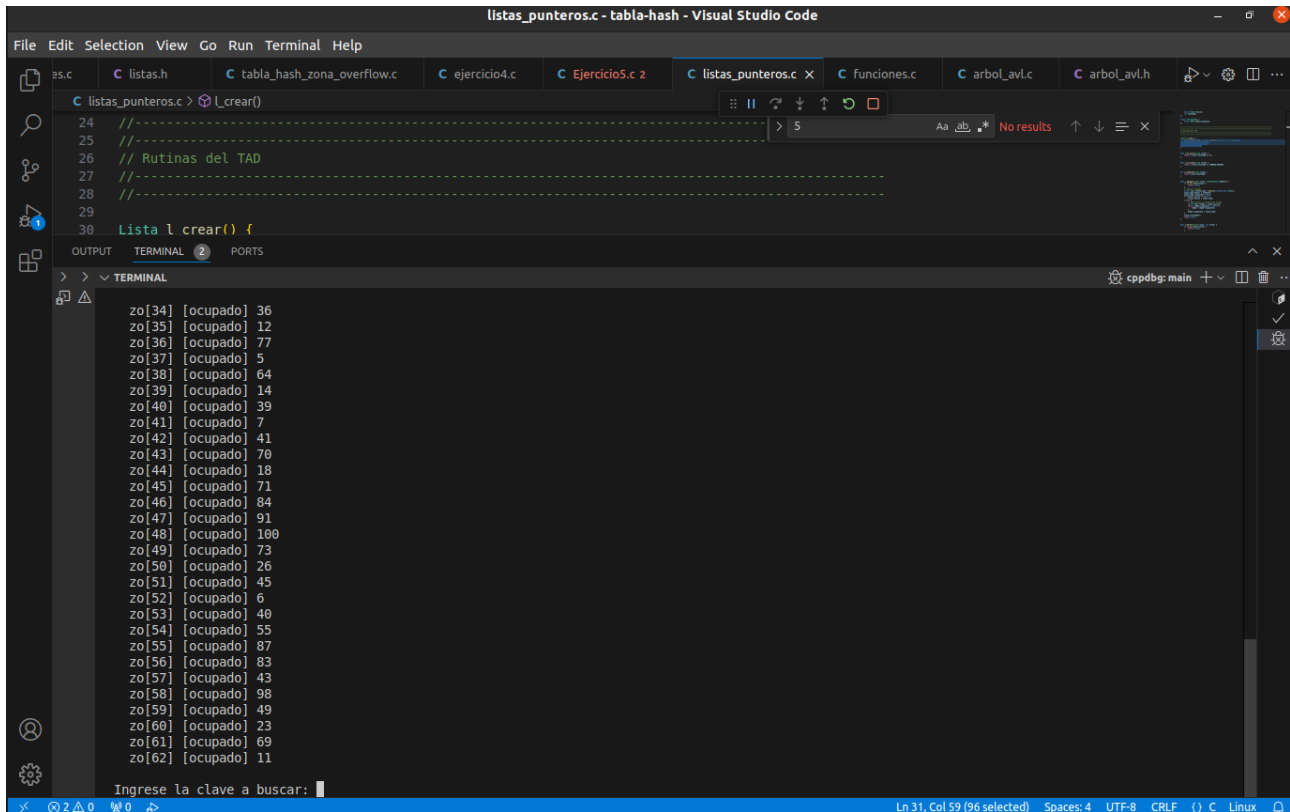
Ingrese la clave a buscar: 1
Ingrese el valor minimo para el rango de claves a buscar: 5
Ingrese el valor maximo para el rango de claves a buscar: 2
Tiempo AVL: s0.000001, Tiempo Hash: s0.000000
Prueba: 2
Ingrese el valor minimo para el rango: 1
```



```
listas_punteros.c - tabla-hash - Visual Studio Code
File Edit Selection View Go Run Terminal Help
...
C listas_punteros.c > ...
24 //-----
25 // Rutinas del TAD
26 //-----
27 //-----
28 //-----
29
30 Lista l crear() {
31     ...
32 }

OUTPUT TERMINAL PORTS
> > > TERMINAL
Tiempo AVL: s0.000001, Tiempo Hash: s0.000000
Prueba: 2
Ingrese el valor minimo para el rango: 1
Ingrese el valor maximo para el rango: 100
Ingrese la cantidad de claves: 2500
Entrada no válida, Ingrese la cantidad de claves: 250
Contenido de la lista: 50 97 24 19 46 47 52 22 56 80 89 65 29 42 51 94 1 35 65 25 15 88 57 44 92 28 66 60 37 33 52 38 29 76 8 75 22 59 96 30 38 36 94
19 29 44 12 29 30 77 5 44 64 14 39 7 41 5 19 29 89 70 18 18 97 25 44 71 84 91 100 73 26 45 91 6 40 55 87 70 83 43 65 98 8 56 5 49 12 23 29 100 44 47 6
9 41 23 12 11 6
Contenido de la tabla hash:
tabla[0] [ocupado] 50
tabla[1] [ocupado] 97
Zona de overflow:
zo[0] [ocupado] 24
zo[1] [ocupado] 19
zo[2] [ocupado] 46
zo[3] [ocupado] 47
zo[4] [ocupado] 52
zo[5] [ocupado] 22
zo[6] [ocupado] 56
zo[7] [ocupado] 80
zo[8] [ocupado] 89
zo[9] [ocupado] 65
zo[10] [ocupado] 29
zo[11] [ocupado] 42
zo[12] [ocupado] 51
zo[13] [ocupado] 94
zo[14] [ocupado] 1
zo[15] [ocupado] 35
zo[16] [ocupado] 25
zo[17] [ocupado] 15
```

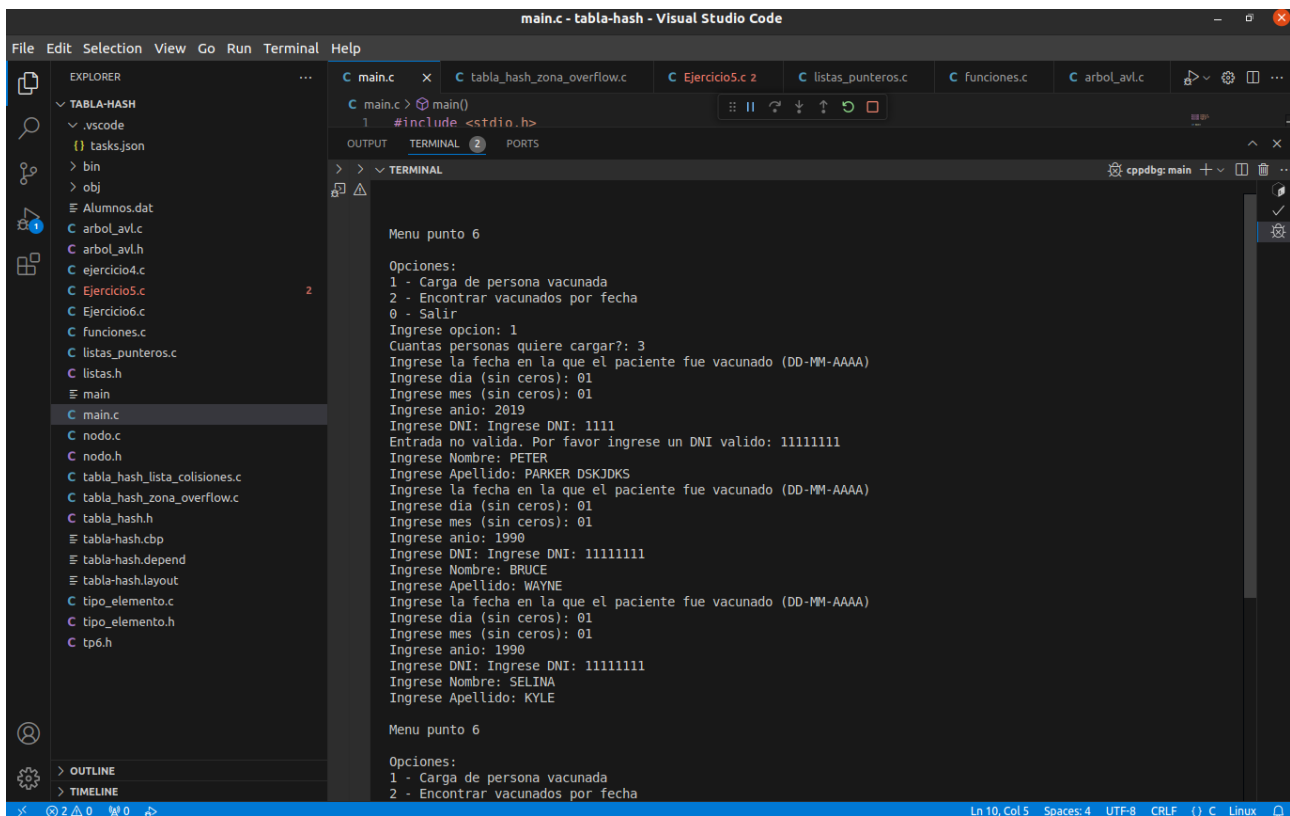
## GRUPO 5 – Correcciones Trabajo Práctico: TABLAS HASH



The screenshot shows the Visual Studio Code editor with the file 'listas\_punteros.c' open. The code defines a function 'l\_crear()' that initializes an array 'zo' of 62 elements, each with a value of 0 and a status of 'ocupado'. The terminal output shows the following data:

Index	Value	Status
zo[34]	36	ocupado
zo[35]	12	ocupado
zo[36]	77	ocupado
zo[37]	5	ocupado
zo[38]	64	ocupado
zo[39]	14	ocupado
zo[40]	39	ocupado
zo[41]	7	ocupado
zo[42]	41	ocupado
zo[43]	70	ocupado
zo[44]	18	ocupado
zo[45]	71	ocupado
zo[46]	84	ocupado
zo[47]	91	ocupado
zo[48]	100	ocupado
zo[49]	73	ocupado
zo[50]	26	ocupado
zo[51]	45	ocupado
zo[52]	6	ocupado
zo[53]	40	ocupado
zo[54]	55	ocupado
zo[55]	87	ocupado
zo[56]	83	ocupado
zo[57]	43	ocupado
zo[58]	98	ocupado
zo[59]	49	ocupado
zo[60]	23	ocupado
zo[61]	69	ocupado
zo[62]	11	ocupado

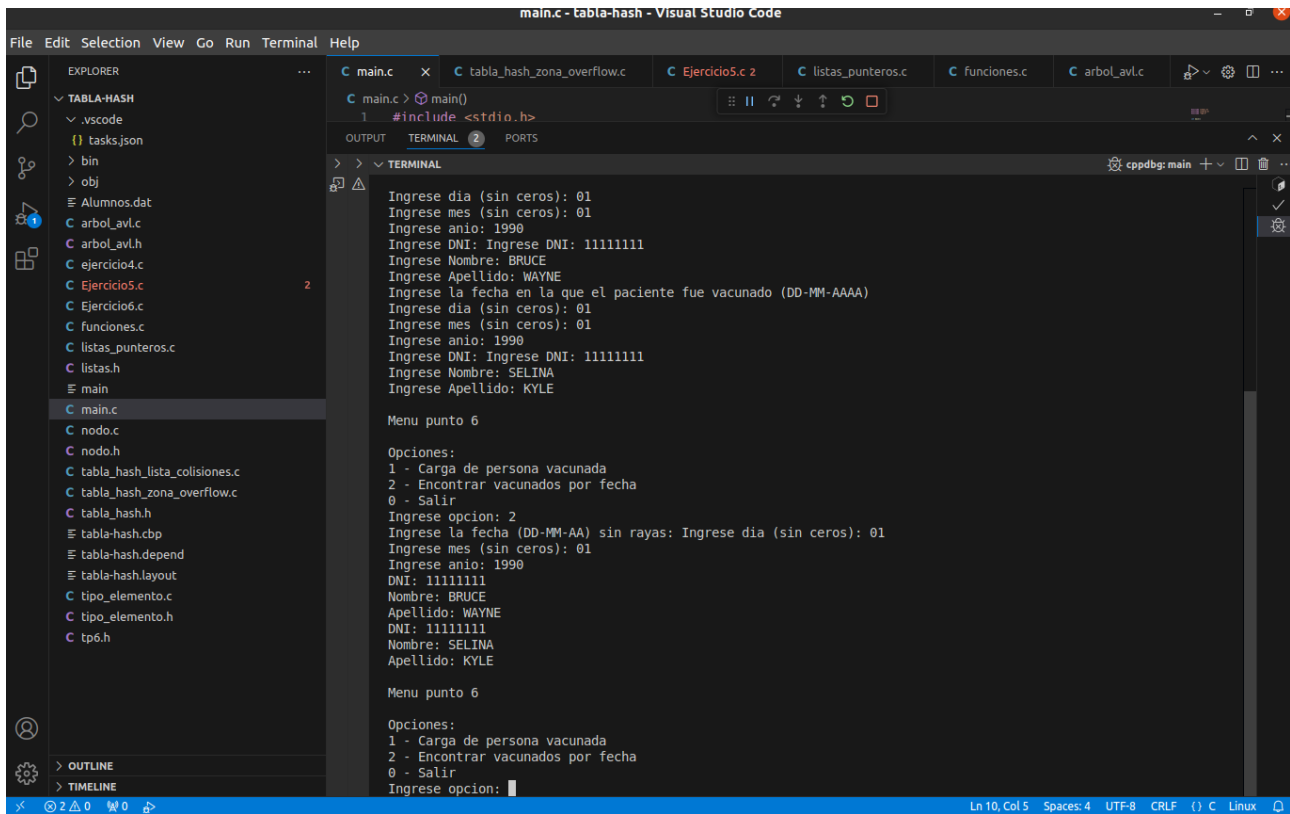
The terminal prompt is 'Ingrese la clave a buscar: '.



The screenshot shows the Visual Studio Code editor with the file 'main.c' open. The code defines a function 'main()' that includes 'stdio.h' and calls 'l\_crear()' from 'listas\_punteros.c'. The terminal output shows the following data:

```
Menu punto 6
Opciones:
1 - Carga de persona vacunada
2 - Encontrar vacunados por fecha
0 - Salir
Ingrese opcion: 1
Cuantas personas quiere cargar?: 3
Ingrese la fecha en la que el paciente fue vacunado (DD-MM-AAAA)
Ingrese dia (sin ceros): 01
Ingrese mes (sin ceros): 01
Ingrese anio: 2019
Ingrese DNI: Ingrese DNI: 1111
Entrada no valida. Por favor ingrese un DNI valido: 11111111
Ingrese Nombre: PETER
Ingrese Apellido: PARKER DSKJDKS
Ingrese la fecha en la que el paciente fue vacunado (DD-MM-AAAA)
Ingrese dia (sin ceros): 01
Ingrese mes (sin ceros): 01
Ingrese anio: 1990
Ingrese DNI: Ingrese DNI: 11111111
Ingrese Nombre: BRUCE
Ingrese Apellido: WAYNE
Ingrese la fecha en la que el paciente fue vacunado (DD-MM-AAAA)
Ingrese dia (sin ceros): 01
Ingrese mes (sin ceros): 01
Ingrese anio: 1990
Ingrese DNI: Ingrese DNI: 11111111
Ingrese Nombre: SELINA
Ingrese Apellido: KYLE
Menu punto 6
Opciones:
1 - Carga de persona vacunada
2 - Encontrar vacunados por fecha
```

## GRUPO 5 – Correcciones Trabajo Práctico: TABLAS HASH



```
main.c - tabla-hash - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  TABLA-HASH
    .vscode
    tasks.json
    bin
    obj
    Alumnos.dat
    arbol_avl.c
    arbol_avl.h
    ejercicio4.c
    Ejercicio5.c
    Ejercicio6.c
    funciones.c
    listas_punteros.c
    listas.h
    main
    main.c
    nodo.c
    nodo.h
    tabla_hash_lista_colisiones.c
    tabla_hash_zona_overflow.c
    tabla_hash.h
    tabla-hash.cbp
    tabla-hash.depend
    tabla-hash.layout
    tipo_elemento.c
    tipo_elemento.h
    tp6.h

main.c
1 #include <stdio.h>

main()
{
    Ingrese dia (sin ceros): 01
    Ingrese mes (sin ceros): 01
    Ingrese anio: 1990
    Ingrese DNI: Ingrese DNI: 11111111
    Ingrese Nombre: BRUCE
    Ingrese Apellido: WAYNE
    Ingrese la fecha en la que el paciente fue vacunado (DD-MM-AAAA)
    Ingrese dia (sin ceros): 01
    Ingrese mes (sin ceros): 01
    Ingrese anio: 1990
    Ingrese DNI: Ingrese DNI: 11111111
    Ingrese Nombre: SELINA
    Ingrese Apellido: KYLE

    Menu punto 6

    Opciones:
    1 - Carga de persona vacunada
    2 - Encontrar vacunados por fecha
    0 - Salir
    Ingrese opcion: 2
    Ingrese la fecha (DD-MM-AA) sin rayas: Ingrese dia (sin ceros): 01
    Ingrese mes (sin ceros): 01
    Ingrese anio: 1990
    DNI: 11111111
    Nombre: BRUCE
    Apellido: WAYNE
    DNI: 11111111
    Nombre: SELINA
    Apellido: KYLE

    Menu punto 6

    Opciones:
    1 - Carga de persona vacunada
    2 - Encontrar vacunados por fecha
    0 - Salir
    Ingrese opcion:

Ln 10, Col 5 Spaces: 4 UTF-8 CRLF () C Linux
```