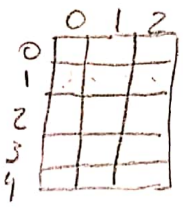
Germán Alvarez Legajo: 21094 DNI: 43157546 LSI Tema 2
Sala 4 Hoja 1

Ejercicio 1

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define F 5
#define C 3

typedef struct {
    char nombre[20];
    int nro;
}meses;

void carga (float xtable[F][C]) {
  int i,j;
  float c;
  printf("Ingrese número de río: ");
  scanf("%d", &i);
  printf("Ingrese numero de mes: ");
  scanf("%d" &j);
  while ((i<F) && (j<C)) {
    printf("Ingrese el caudal: ");
    scanf("%f", &c);
    xtable[i-1][j-1] = c;
    printf("Ingrese número de río: ");
    scanf("%d", &i);
    printf("Ingrese numero de mes: ");
    scanf("%d", &j);
  }
  return;
}

void cargarMeses(meses m[C]){
  for(int i=0; i<C; i++) {
      m.nro = i+1;
      printf("Ingrese nombre: ");
      gets(m.nombre);
  }
  return;
}

float promedio (float xtable[F][C]) {
float p;
for(int j=0;j<C;j++) {
  p += xtable[1][j];
}
return p;
}

void indicar (float xtable[F][C], char xmes[20], meses mm) {
  int i=0,m; float max=0;
  while((i<C) && (strcmp(xmes, mm[i].nombre) != 0)) {
      i++;
  }
  for(int j=0; j<F; j++) {
      if (max < xtable[j][i]) {
          max = xtable[j][i];
          m = j;
      }
  }
  printf("El rio con el mayor caudal es %d:", m);
```



Escaneado con CamScanner

```
    return;
}
int main (void) {
  float tabla[F][C];
  meses M[C];
  char nombre[20];
  carga(tabla);
  cargaMeses(M);
  printf("\nEl promedio de caudal de río Jáchel es: %.f ", promedio(tabla));
  printf("\nIngrese nombre de un mes: ");
  gets(nombre);
  indicar(tabla, nombre, M);
  return 0;
}
```

Ejercicio 2

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
struct inqui {
  float importe;
  char nom[40], cel[12];
  int nrod, pago;  /*(1-pago, 2-no pago)*/
};
/*prototipos de funciones*/
void punto_a(FILE *archi);
void punto_b(FILE *archi);
void punto_c(FILE *archi, puntero &cab);

int main (void) {
  FILE *archivo;
  puntero cabeza;
  int numero;
  archivo = fopen("Inquilinos.dat", "a+");
  if(archivo != NULL) {
      punto_a(archivo);
      punto_b(archivo);
      punto_c(archivo, cabeza);
  } else {
    printf("Hubo error en la apertura del archivo");
  }
  return 0;
}
```

```
struct liste {
  char nom[40];
  char cel[12];
  int nrod;
  struct liste *sig;
};
typedef struct liste * puntero;
```

```
printf("\nIngrese numero de deportonado: ");
scanf("%d ", &numero);
punto_d(cabeza, numero);
```

Escaneado con CamScanner

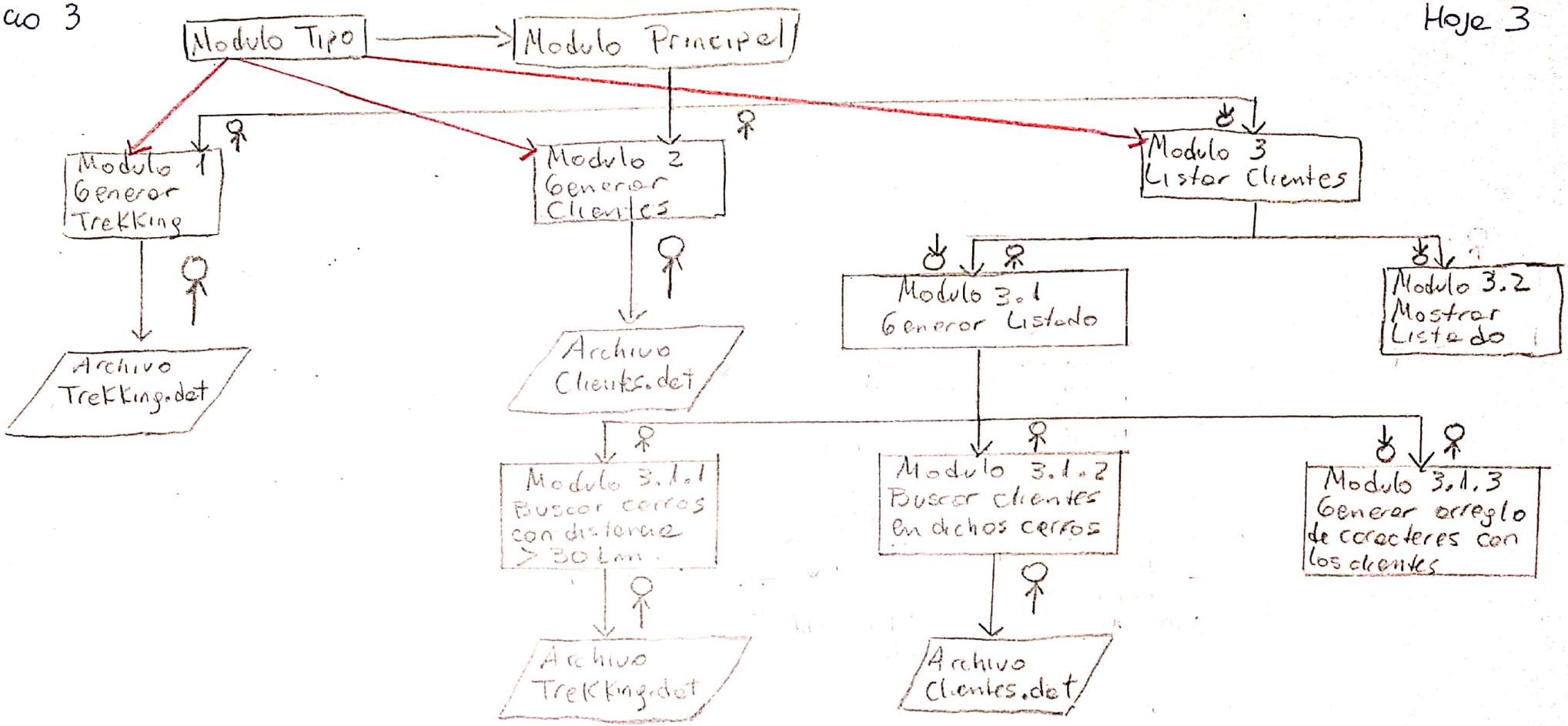
```
void punto_a (FILE *archi) {
char nombre[40];
inqui puntoA;
rewind(archi);
printf("\nIngrese nombre del inquilino:");
gets(nombre);
fread(&puntoA, sizeof(puntoA), 1, archi);
while((!feof(archi)) && (strcmp(nombre, puntoA.nom) != 0)) {
   fread(&puntoA, sizeof(puntoA), 1, archi);
}
if (!feof(archi)) {
   printf("El importe actual es: %.f", puntoA.importe);
   printf("\nIngrese nuevo importe:");
   scanf("%f", &puntoA.importe);
   fseek(archi, -sizeof(puntoA), SEEK_CUR);
   fwrite(&puntoA, sizeof(puntoA), 1, archi);
}
return;
}
void punto_b (FILE *archi) {
  inqui puntoB;
  rewind(archi);
  while((fread(&puntoB, sizeof(puntoB), 1, archi) != 0)) {
     if (puntoB.pago == 2) {
        printf("El inquilino %s debe en alquiler %f", puntoB.nom, puntoB.importe);
     }else{
        printf("El inquilino %s no debe alquiler ", puntoB.nom);
     }
  }
  return;
}
void punto_c (FILE *archi, puntero &cab) {
  puntero nuevo; -> inqui puntoC;
  rewind(archi);
  cab = NULL;
  while((fread(&puntoC, sizeof(puntoC), 1, archi) != 0)) {
       if(puntoC.pago == 1) {
        nuevo=(puntero) malloc(sizeof(struct lista));
        strcpy(nuevo->nom, puntoC.nom);
        strcpy(nuevo->cel, puntoC.cel);
        nuevo->nrod = puntoC.nrod;

        nuevo->sig = cab;
        cab = nuevo;
        }
  }
  return;
```

Escaneado con CamScanner

```
void punto_d(puntero *cab, int xnumero) {
  puntero ant, p;
void eliminar(puntero p, puntero ant, int nro); /* prototipo de funcion */
  if (cab->nrod == xnumero) {
        p = cab;
        cab = cab->sig;
        free(p);
  }else{
      p=cab, ant=cab;
      eliminar(p,ant, xnumero);
  }
  return;
}
void eliminar(puntero p, puntero ant, int xnro) {
   if((p!=NULL) && (p->nrod != xnro)) {
         anterior = p;
         p = p->sig;
         eliminar(p, ant, xnro);
    }else{
        if(p!=NULL) {
          printf("\n Eliminamos inquilino %s", p->nom);
           free(p);
        }else{
          printf("\n No se pudo eliminar inquilino\n");
        }
    }
 return;
}
```

Ejercicio 3 en la sig. hoja (Hoja 3)

Escaneado con CamScanner



Modulo Principal : Cohesión : Secuencial , Acoplamiento : Estampado
Modulo Tipo : Cohesion : Abstraccional , Acoplamiento : Estampado
Modulo 1 : Cohesion : Comunicación , Acoplamiento : Externo
Modulo 2 : Cohesion : Comunicación , Acoplamiento : Externo
Modulo 3 : Cohesion : Secuencial , Acoplamiento : Estampado
Modulo 3.1 : Cohesion : Secuencial , Acoplamiento : Estampado
Modulo 3.1.1 : Cohesion : Secuencial , Acoplamiento : Externo
Modulo 3.1.2 : Cohesion : Secuencial , Acoplamiento : Externo
Modulo 3.1.3 : Cohesión : Comunicación , Acoplamiento : Normal
Modulo 3.2 : Cohesion : Comunicación , Acoplamiento : Normal

Escaneado con CamScanner