Examen de Lenguajes de Programación

4 de Julio del 2002 - 2 horas

Pregunta 1 (30 puntos)

Se tiene una representación matricial de los departamentos de un edificio, ordenados por piso. El edificio cuenta con 14 pisos y cada piso cuenta con un máximo de 6 departamentos.

Para cada departamento, se almacenan exclusivamente los siguientes datos: el nombre del dueño, la superficie del departamento (en m^2) y si éste cuenta o no con bodega.

Los "gastos comunes" mensuales que se debe pagar por cada departamento a la administración del edificio, corresponden a \$350 por cada m^2 de superficie + \$2000 si el departamento cuenta con bodega.

1. Defina las estructuras de datos a utilizar

   ```
   struc building
   {
   char dueno[20];
   int superf;
   int bod;
   }

   5 puntos si incluyeron sólo estos 3 campos.

   2 puntos si sobran o faltan campos.
   ```

2. Escriba una función en C que, recibiendo como parámetro sólo la matriz del edificio, permita conocer el nombre del dueño y el número del o los departamentos que pagan una mayor cantidad por concepto de gastos comunes. La función debe permitir conocer dicha cantidad. Ejemplos de número de departamento en este edificio son:

   - Primer departamento Piso 1 : 101
   - Segundo departamento Piso 1 : 102

- Tercer departamento Piso 8 : 803
- etc... (No existe Piso Cero)

```
10 puntos para el calculo de maximo gastos comunes
(solo 5 puntos si no lo hacen para un maximo de 6 departamentos,
recuerden que pueden haber menos de 6 departamentos por piso)

10 puntos por impresion de resultados incluyendo
el calculo de numero de departamentos (5 puntos si no)

5 puntos por definicion de funcion y sintaxis.

void gastos_comunes(struct building edif[][])
{
int i,j, gc[14][6], max=0, num_depto;
for (i=0; i<14; i++)
    for (j=0 ; j<6 ; j++)
        if (edif[i][j].superf != 0)
        {
        /* superficie */
        gc[i][j] = edif[i][j].superf*350;

        /* bodega */
        gc[i][j] = gc[i][j]+edif[i][j].bod*2000;

        /* es el maximo? */
        if (gc[i][j] > max)
            max = gc[i][j];
        };

printf ("Los gastos comunes más altos ");
printf ("son $%d, pagados por:\n", max);

for (i=0; i<14; i++)
    for (j=0 ; j<6 ; j++)
        if (gc[i][j] == max)
        {
        num_depto = (i+1)*100 + (j+1);
        printf ("Departamento %d\t", num_depto);
        printf ("Dueño: %s", edif[i][j].dueno);
        };
}
```

Pregunta 2 (20 puntos)

Suponga que tiene un archivo de números enteros, implemente la función recursiva `void funcion(FILE *ARCHIVO)` que imprime en pantalla el contenido del archivo en orden inverso (usted no sabe como se llama el archivo, sólo tiene su descriptor). Ejemplo:

archivo: `1 3 2712 14 16 18 205 7 9 22 25`

salida: `25 22 9 7 205 18 16 14 2712 3 1`

Nota: No se preocupe de los espacios en blanco ni los saltos de línea.

```
5 puntos por leer del archivo.
5 puntos por el caso base.
10 puntos por la recursión (primero llamo, despues imprimo)

funcion(FILE *ARCHIVO)
{
int i;

/* leo */
fscanf (ARCHIVO,"%d",&i);

/* caso base: archivo vacio */
if (feof(ARCHIVO)) return;

/* recursion */
funcion (ARCHIVO);
printf ("%d ",i);
}
```

Pregunta 3 (30 puntos)

Para la siguiente estructura:

```
struct nodo {
  int dato;
  struct nodo *sigte;
};
```

Implemente la función: `struct nodo * elimina(struct nodo *primero, int eliminado);` que elimina los nodos de la lista apuntada por "primero", cuyo valor corresponde al de la

variable “eliminado”. Debe retornar un puntero al primer nodo de la lista modificada.

Nota: Si no se encuentra un nodo cuyo valor sea igual al de la variable “eliminado”, la lista no debe modificarse.

```
struct nodo *elimina(struct nodo *primero, int eliminado) {
   struct nodo *p, *q;

   /* si el dato del primero debe ser eliminado ... */
   while(primero != NULL && primero->dato == eliminado) {
     p = primero;

     /* ... se enlaza al nuevo primero ... */
     primero = primero->sigte;

     /* ... y se elimina el anterior */
     free(p);
   }

   /* en caso de lista no vacía */

   if(primero != NULL) {

     /* par de variables para recorrer, "p" y "q", donde "q"
     va tras de "p" */

     p = primero->sigte;
     q = primero;

     /* mientras no se acabe la lista */

     while(p != NULL) {
       /* si el nodo apuntado por "p" debe eliminarse ... */

       if(p->dato == eliminado) {

        /* ... se enlaza "q", anterior a "p", con el siguiente a
           "p" ... */
        q->sigte = p->sigte;

        /* ... y se elimina el nodo */
        free(p);
```

```
       } else {

        /* sino, "q" avanza hacia "p" ... */
        q = p;
       }

       /* ... y "p" se sitúa delante de "q" */
       p = q->sigte;
     }
   }
return primero;
}
```

Puntaje:

```
 * identificar el primer nodo a eliminar (5)
 * identificar el último que (o el primero que no) se debe
eliminar (5)
 * eliminación propiamente tal de cada uno de los iguales (10)
 * correcto enlace de las elementos de lista (10)
```

Pregunta 4 (20 puntos)

Suponga que una cola de enteros está definida como:

```
struct fila
{
pila p1;
pila p2;
};
typedef struct fila cola;
```

Donde `p1` y `p2` son pilas de enteros. Implemente las funciones `void encolar (cola c, int x)` e `int decolar (cola c)` utilizando solamente las funciones de pilas `void poner (pila p, int x)`, `int sacar (pila p)` e `int vacia(pila p)` y **SOLO UNA** variable auxiliar.

Nota: Lo importante de esta pregunta es el correcto uso de las funciones, no la forma en que está implementada la pila.

```
5 puntos por encolar y 15 por decolar
(0 puntos en decolar si usa mas de una variable auxiliar)
```

```
void encolar (cola c, int x)
{
/* 5 puntos */
poner(c.p1,x);
}

int decolar (cola c)
{
int aux;

/* 5 puntos por vaciar una pila en la otra */
while (!vacia(c.p1))
    poner(c.p2,sacar(c.p1));

/* 5 puntos por guardar y retornar lo pedido */
aux = sacar(c.p2);

/* 5 puntos por retornar todo al estado original */
while (!vacia(c.p2))
    poner(c.p1,sacar(c.p2));

/* retornar!!! */
return aux;
}
```

Puntajes analogos si encolar se lleva el peso de la ejecucion.