


Segundo Parcial de Programación Imperativa (72.31)

14/06/2019

	Ejercicio 1	Ejercicio 2	Ejercicio 3	Nota	Firma Docente
Entregado	/	/	/	----	
Calificación	0 /4	0 /4	1 /2	1	

- ❖ **Condición mínima de aprobación: Sumar 5 puntos**
- ❖ Se tendrá en cuenta en la calificación el **ESTILO** y la **EFICIENCIA** de los algoritmos.
- ❖ Los ejercicios que no se ajusten estrictamente al enunciado, **no serán aceptados**.
- ❖ Puede entregarse en lápiz.
- ❖ No usar variables globales ni static.
- ❖ En caso de necesitar usar malloc o similar, no validar que retorne distinto de NULL.
- ❖ No es necesario escribir los #include.
- ❖ Escribir en cada hoja Apellido, Legajo, Número de hoja y Total de hojas entregadas.
- ❖ Realizar los ejercicios en hojas separadas.

Ejercicio 1

Dada una lista de enteros cuya definición de tipos es la siguiente:

```
typedef struct node * TList;

typedef struct node {
    int elem;
    struct node * tail;
} TNode;
```

y donde una lista vacía se representa con el valor NULL.

- a) Escribir una función **recursiva removeOdd** que, recibiendo únicamente una lista, **elimine de la misma** todos los elementos en **posición impar** (asumiendo que el primer elemento de la misma está en la posición par 0). No utilizar funciones auxiliares.

Lista Original	Debe quedar
1 -> 2 -> 3 -> 4 -> 5 -> NULL	1 -> 3 -> 5 -> NULL
6 -> 7 -> 8 -> 9 -> NULL	6 -> 8 -> NULL
10 -> NULL	10 -> NULL (Sin cambios)
NULL	NULL (Sin cambios)

- b) Escribir una función **recursiva removeRepeated** que, recibiendo únicamente una lista ordenada en forma no descendente, **elimine de la misma** los **elementos repetidos**. No utilizar funciones auxiliares.

Lista Original	Debe quedar
1 -> 1 -> 1 -> 2 -> 2 -> NULL	1 -> 2 -> NULL
6 -> 7 -> 8 -> 9 -> NULL	6 -> 7 -> 8 -> 9 -> NULL (Sin cambios)

Ejercicio 1

a. Eliminar elementos en posición impar

```
int removeOdd (Tlist list){  
    if (list == NULL || list -> tail == NULL)  
        return list;  
    Tlist aux = list -> tail -> tail;  
    list -> tail = aux;  
    list -> tail = removeOdd (list -> tail);  
    return list;  
}
```

b. Eliminar repetidos

```
Tlist removeRep (Tlist list){  
    if (list == NULL || list -> tail)  
        return list;  
    list -> tail = removeRep (list -> tail);  
    if (list -> elem == list -> tail -> elem){  
        Tlist aux = list -> tail -> tail;  
        list -> tail = aux;  
    }  
    return list;  
}
```

Ejercicio 2

Se desea guardar una colección de elementos no repetidos, en la cual los elementos más "populares" (los que más se consultan) estén al principio de la colección. De esta forma, será más rápido acceder a los elementos que más veces se consulten. Para ello se definió que el conjunto de datos opere de la siguiente forma:

- Cuando se inserta un elemento (no repetido) se lo inserta **al final**
- Cuando se consulta un elemento (con la función `get`) el mismo es "swapeado" con el elemento anterior.

El contrato con el TAD es el siguiente:

`popularSetADT.h`

```
typedef struct popularSetCDT * popularSetADT;

typedef ... elemType; // Tipo de elemento a insertar

/*
** Retorna 0 si los elementos son iguales, negativo si e1 es "menor" que e2 y positivo
** si e1 es "mayor" que e2
*/
static int compare (elemType e1, elemType e2) {
    ...
}

/* Retorna un nuevo conjunto de elementos genéricos. Al inicio está vacío */
popularSetADT newPopularSet(?);

/* Inserta un elemento si no está. Lo inserta al final.
** Retorna 1 si lo agregó, 0 si no.
*/
unsigned int add(popularSetADT popularSet, elemType elem);

/* Retorna la cantidad de elementos que hay en la colección */
unsigned int size(popularSetADT popularSet);

/* Se ubica al principio del conjunto, para poder iterar sobre el mismo */
void toBegin(popularSetADT popularSet);

/* Retorna 1 si hay un elemento siguiente en el iterador, cero si no */
int hasNext(popularSetADT popularSet);

/* Retorna el siguiente elemento. Si no hay siguiente elemento, aborta */
elemType next(popularSetADT popularSet);

/* Retorna una copia del elemento. Si no existe retorna NULL.
** Para saber si el elemento está, usa la función compare.
** Si el elemento estaba y no es el primero, lo intercambia con el anterior
*/
elemType * get(popularSetADT popularSet, elemType elem);
```

Donde `?` en una lista de parámetros indica que usted (programador) debe definir cuáles son los parámetros necesarios para esa función, en base a las características del TAD.

Implementar el TAD completo (el archivo `popularSetCDT.c`).

Ejemplo de uso, considerando los siguientes cambios en popularSetADT.h

```
typedef struct {
    int code;
    char name[20];
} elemType;

static int compare (elemType e1, elemType e2) {
    return e1.code - e2.code;
}
```

```
#include "popularSetADT.h"
```

```
int
main(void) {
    popularSetADT p = newPopularSet(...); // completar parametros si es necesario
    elemType aux = {1, "uno"};
    add(p, aux); // retorna 1
    strcpy(aux.name, "dos");
    add(p, aux); // retorna 0
    p.code = 2;
    add(p, aux); // retorna 1
    p.code = 3;
    add(p, aux); // retorna 1
    p.code = 4;
    add(p, aux); // retorna 1
    toBegin(p);
    while (hasNext(p)) {
        aux = next(p);
        printf("%d ", aux.code);
    }
    putchar('\n');

    aux.code = 5;
    elemType * q = get(p, aux); // retorna NULL

    aux.code = 4;
    q = get(p, aux);
    free(q);
    q = get(p, aux);
    free(q);
    aux.code = 3;
    q = get(p, aux);
    free(q);
    toBegin(p);
    while (hasNext(p)) {
        aux = next(p);
        printf("%d ", aux.code);
    }
    putchar('\n');
    return 0;
}
```

Al ejecutar el programa la salida será:

```
1 2 3 4
1 4 3 2
```

Ejercicio 2

```
struct popularSetADT{
    size_t size;
    elemType * vec;
    int next;
}

popularSetADT newPopularSet (void){
    return calloc(1, sizeof(struct popularSetADT));
}

void toBegin (popularSetADT set){
    set->next = 0;
}

int hasNext (popularSetADT set){
    return (set->next) != set->size;
}

elemType next (popularSetADT set){
    if (!hasNext(set))
        exit(1);
    elemType aux = set->vec[set->next];
    set->next += 1;
    return aux;
}

unsigned int size (popularSetADT set){
    return set->size;
}

unsigned int add (popularSetADT set, elemType elem){
    int i;
    for (i = 0; i < set->size; i++){
        if (compare (set->vec[i], elem) == 0)
            return 0;
    }
    if (i % BLOQUE == 0)
        set->vec = realloc (set->vec, (i + BLOQUE) * sizeof(elemType));
    (set->size)++;
    set->vec[i] = elem;
    return 1;
}
```

```

elemType * get (popularSetADT set, elemType elem){
    int i;
    for(i=0, i < set->size, i++){
        if(compare (set->vec[i], elem) == 0){
            elemType * aux = malloc(sizeof (elemType));
            *aux = set->vec[i-1];
            set->vec[i-1] = set->vec[i];
            set->vec[i] = *aux;
            return aux;
        }
    }
    return NULL;
}

```


Ejercicio 3

- a) Marcar las líneas que producen **errores de compilación**. Explicar por qué se produce el error y cómo corregirlo.

```

1  #include <stdio.h>
2  #define DIM 20
3
4  typedef struct {
5      int a;
6      int b;
7  } Coordenada;
8
9  typedef Coordenada * PCoord;
10
11 typedef struct {
12     char nombre[DIM];
13     Coordenada punto1;
14     PCoord punto2;
15 } TipoX;
16
17 int
18 main(void) {
19     TipoX var1;
20     var1.nombre = "nombre1";
21     var1.punto1.a = 10;
22     var1.punto1.b = 30;
23     var1->punto2.a = 20;
24     var1->punto2.b = 50;
25     return 0;
26 }

```

- b) Indicar la **salida estándar** de los siguientes programas. Justificar.

Programa 1	Programa 2
<pre> #include <stdio.h> #define MAX 50 typedef struct { int codigo; char descripcion[MAX]; } Artículo; void cambia(Artículo art) { art.codigo = 10; art.descripcion = "Copia"; } int main(void){ Artículo art = {1, "Original"}; cambia(art); printf("%d %s\n", art.codigo, art.descripcion); return 0; } </pre>	<pre> #include <stdio.h> #include <string.h> #define MAX 50 typedef struct { int codigo; char descripcion[MAX]; } Artículo; void cambia(Artículo art) { art.codigo = 10; strcpy(art.descripcion, "Copia"); } int main(void) { Artículo art = {1, "Original"}; cambia(art); printf("%d %s\n", art.codigo, art.descripcion); return 0; } </pre>

Ejercicio 3

0. Lineas z0, z3 y z4

- Error pues nombre[DIN] es vector estatico
- var1.puntoz -> a = z0;
- var1.puntoz -> b = 50;

b. Indicor salida estandar

1. Error de ejecucion

2. Imprime : 1 Original

