EXAMEN FINAL AEDD - ISI-FRSF-UTN 4/2/2019 Alumno:

1) La función Uno( ) recibe dos arreglos de 500 números enteros (A y B), los tamaños lógicos de cada uno (TA y TB, TB <= TA) y un valor booleano (C).
El arreglo A y el arreglo B están desordenados.
La función Uno() debe retornar un vector V y su tamaño lógico TV, ordenado ascendentemente si C==True (descendentemente en otro caso) conteniendo solamente los valores de B que se encuentran en A, sin repeticiones. Los vectores A y B pueden contener elementos repetidos.
La función Uno() debe retornar, además, la cantidad de elementos de A (no se cuentan los repetidos) que no fueron encontrados en B.
Codificar la cabecera y el cuerpo de la función Uno( ).

---

2) Defina los datos necesarios y escriba una función para resolver el siguiente problema:
La matriz M de 12 columnas por 8 filas, contiene datos del año pasado: el total de ventas mensuales de 8 sucursales de una red de perfumerías con presencia en toda la provincia.
Codifique una función Dos( ), que reciba esta matriz y evalúe:

- Informe si hubo alguna sucursal que registró ventas individuales cada mes mayores que las otras sucursales durante todo el año.
- Determine cuáles fueron los dos meses con menor total de ventas.

---

3) Defina los datos necesarios y escriba una función para resolver el siguiente problema:

- Un sistema de alertas por inundaciones almacena un registro diario (para los últimos 30 días), con la siguiente información en cada nodo:
  - Altura río en puerto local: valores en 0.0 a 10.0 (altura en mts.)

Codifique una función que reciba una lista enlazada simple L, con esta información en cada uno de sus nodos y:

- Retorne la diferencia entre la altura registrada en el último nodo con respecto a la altura que había el primer día del mes.
- Informe si hay posibilidades de “inundación repentina” (en los últimos 5 días, se dieron al menos 3 días consecutivos de creciente y tuvo un incremento de más de 1 mt en el total de los 5 días).

---

4) **Codificar una función recursiva** Ejercicio4(…) que reciba un vector V de enteros, su tamaño lógico TL y un valor entero Aux, y retorne la cantidad de veces que la suma entre dos elementos consecutivos del vector es igual a Aux.
Ej.: Si TL=8, V={5, 2, 8, 9, 6, 3, 12, 0} y Aux=15, la llamada a la función Ejercicio4(....) debe retornar 2.

5- a) Qué mecanismos de paso de parámetros existen en C++? Explique brevemente sus diferencias.
b) Cuál es la complejidad de la búsqueda binaria de un elemento en un vector ordenado? y la complejidad de la búsqueda de un elemento en una lista ordenada?
c) Se necesitan almacenar los datos de 500000 afiliados a una obra social (un struct de sizeof = 350 bytes) en un archivo. Ud. utilizaría un archivo de texto o un archivo binario?. Fundamente.
d) Dar el valor de verdad de las siguientes afirmaciones y justificar la respuesta:
i) La operación de Quitar el primer elemento de una lista, una pila o una cola, tiene siempre la misma complejidad.
ii) Borrar un elemento de un vector ordenado tiene complejidad constante.
iii) Insertar un elemento en una lista ordenada tiene complejidad lineal.

Puntaje: Ej. 1.: 24ts, Ej. 2: 22 pts, Ej. 3: 22 pts Ej4: 12 pts Ej. 5: 20 (5 c/u)

Soluciones

Ejercicio 1:

Ejercicio 2:

```
#define FILAS 8
#define COLUMNAS 12
float M[FILAS][COLUMNAS];

// Funcion con el primer punto solicitado
void Dos_1(float matriz[][COLUMNAS]){
	// Arreglo para determinar sucursal con más ventas
	bool sucursalMasVentas[FILAS] = {true, true, true, true, true, true, true, true};
	// Variable de corte para while
	bool variableDeCorte = false;

	for(int j=0; j<COLUMNAS; j++){ // Por cada mes
		for (int i = 0; i < FILAS; i++){ // Por cada sucursal
			// Se compara una sucursal en un mes con el resto de las
sucursales el mismo mes
			// (solo si aún no se determinó que NO es la sucursal con más
ventas)
			if(sucursalMasVentas[i]==true){
				int k = 0;
				variableDeCorte = false;
				// Se usa un while para establecer un corte previo sin usar
break.
				while(k<FILAS && !variableDeCorte){   // Se compara una
sucursal en un determinado mes, con todas las sucusales restantes.
					// Si ocurre que las ventas son menores que en otra
sucursal en dicho mes,
					// se determina que dicha sucursal no es la que más
vendió y el while termina.
					if(k != i && matriz[i][j] <= matriz[k][j]){
						sucursalMasVentas[i] = false;
						variableDeCorte = true;
					}
					k++;
				}
			}
		}
	}

	// Se muestra (si existe) la sucursal que más vendió
	int i = 0;
	while (sucursalMasVentas[i] != true && i<=8){
		i++;
	}
```

```
        if(i<8){
                cout << "La sucursal " << i+1 << " registro ventas individuales cada mes
mayores que las otras sucursales durante todo el año." << endl;
        }
        else{
                cout << "Ninguna sucursal registro ventas individuales cada mes
mayores que las otras sucursales durante todo el año." << endl;
        }
}


// Funcion con el segundo punto solicitado
void Dos_2(float matriz[][COLUMNAS]){
        float min1, min2;
        int mes1, mes2;

        // Arreglo para determinar total de ventas por mes
        float totalVentasXMes[COLUMNAS] = {0, 0, 0, 0, 0, 0, 0, 0};

        for(int j=0; j<COLUMNAS; j++){ // Por cada mes
                for (int i = 0; i < FILAS; i++) // Se obtiene el total de ventas
                        totalVentasXMes[j] = totalVentasXMes[j] + matriz[i][j];
        }

        // Se toman los 2 primeros meses y con ellos se inicializan 2 variables
        if (totalVentasXMes[0] < totalVentasXMes[1]){
                min1=totalVentasXMes[0];
                mes1=0;
                min2=totalVentasXMes[1];
                mes2=1;
        }
        else {
                min1=totalVentasXMes[1];
                mes1=1;
                min2=totalVentasXMes[0];
                mes2=0;
        }
        // Se comparan los valores de los meses restantes
        // Se guardan los totales más chicos y los meses correspondientes
        for(int j=2; j<COLUMNAS; j++){
                if (totalVentasXMes[j] <= min1){
                        min2=min1;
                        mes2=mes1;
                        min1=totalVentasXMes[j];
                        mes1=j;
                }
```

```
                else if (totalVentasXMes[j] <= min2){
                        min2=totalVentasXMes[j];
                        mes2=j;
                }
        }
        cout << "Los 2 meses con menor total de ventas son: " << mes1+1 << " y " <<
mes2+1<< endl;
}
```

Ejercicio 3:

```
struct Nodo {
	float altura;
	Nodo* sig;
};
typedef Nodo* nodoPtr;

float diferenciaAlturasMes(nodoPtr lista) {
	nodoPtr ultimo;
	for(ultimo = lista; ultimo->sig != NULL; ultimo = ultimo->sig);
	return ultimo->altura - lista->altura;
}

bool hayPosibilidadDeInundacion(nodoPtr lista) {
	nodoPtr dia26 = lista;
	for(int i = 1 ; i<26; i++) {
		dia26 = dia26 -> sig;
	}

	nodoPtr anterior = dia26;
	nodoPtr actual = dia26->sig;

	bool crecienteConfirmada = false;
	float altura26 = dia26 -> altura, diasCreciendo = 0;

	while(actual != NULL) {
		if(actual->altura > anterior->altura) {
			diasCreciendo++;
		} else {
			diasCreciendo = 0;
		}
		if(diasCreciendo == 3) crecienteConfirmada = true;
		anterior = actual;
		actual = actual->sig;
	}
	float altura30 = anterior -> altura;
	return altura30 - altura26 > 1 && crecienteConfirmada;
}

float ejercicio3(nodoPtr lista) {
	if(hayPosibilidadDeInundacion(lista))
		cout << “Hay posibilidad de inundación repentina” << endl;
	return diferenciaAlturasMes(lista);
}
```

Ejercicio 4:

```
int sumaDosConsecutivos(int V[], int TL, int aux) {
        if(TL < 2) return 0;
        return (V[TL-1] + V[TL-2]==aux) + sumaDosConsecutivos(V, TL-1, aux);
}
```

Ejercicio 5:

a) Qué mecanismos de paso de parámetros existen en C++? Explique brevemente sus diferencias.
En C++ hay dos mecanismos de paso de parámetros principales: por copia y por referencia.
En el mecanismo de paso de parámetros por copia (también llamado “por valor”), al momento de la llamada se crea una copia del valor de los parámetros actuales (aquellos utilizados efectivamente en la invocación) y la función llamada recibe y trabaja sobre dichas copias. Al retorno de la función la memoria utilizada para dichos parámetros se libera, y si fueron modificados, dichos cambios no afectan el valor de los parámetros actuales.
En el mecanismo de paso de parámetros por referencia, al momento de la llamada lo que se pasa a la función llamada es la dirección del parámetro. La función invocada recibe dicha dirección y con ella puede acceder y modificar si es necesario el valor de los parámetros actuales. Los cambios efectuados por la función invocada sobre los parámetros pasados por referencia, permanecerán luego del retorno desde dicha función.
Algunos tipos de datos pasan por default como parámetros por copia: int, float, bool, etc; otros, como los vectores, pasan por default como parámetros por referencia.

b) Cuál es la complejidad de la búsqueda binaria de un elemento en un vector ordenado? y la complejidad de la búsqueda de un elemento en una lista ordenada?

- buscar un elemento en un vector ordenado con una búsqueda binaria tiene complejidad logarítmica.
- buscar un elemento en una lista ordenada tiene complejidad lineal.

c) Se necesitan almacenar los datos de 500000 afiliados a una obra social (un struct de sizeof = 350 bytes) en un archivo. Ud. utilizaría un archivo de texto o un archivo binario?. Fundamente.

- Es una cantidad considerable de registros, todos tienen la misma longitud en bytes. Generalmente con archivos de esas características, luego se precisa acceder a los mismos en algún orden (por apellido, por número de afiliado, etc.), los registros podrían estar almacenados en ese orden, y eso optimizaría las búsquedas de afiliados en particular. Por esas razones utilizaría un archivo binario.

d) Dar el valor de verdad de las siguientes afirmaciones y justificar la respuesta:

i) La operación de Quitar el primer elemento de una lista, una pila o una cola, tiene siempre la misma complejidad.

- Verdadero, en cualquiera de los tres casos es siempre una cantidad constante de operaciones.

ii) Borrar un elemento de un vector ordenado tiene complejidad constante.

- Falso, borrar un elemento de un vector ordenado tiene complejidad lineal.

iii) Insertar un elemento en una lista ordenada tiene complejidad lineal.

- Verdadero (si consideramos que para Insertar, necesitamos primero encontrar en qué lugar de la lista se tiene que realizar la inserción)
- Falso (si consideramos que ya tenemos apuntada la posición en la cual vamos a insertar el elemento, en cuyo caso la Inserción tiene complejidad constante).