

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30



Esta prueba solo pueden realizarla los estudiantes que hayan aprobado la evaluación continua

Este enunciado también corresponde a las siguientes asignaturas:

- 81.578 - Prácticas de programación

Ficha técnica de la prueba

- No es necesario que escribas tu nombre. Una vez resuelta la prueba final, solo se aceptan documentos en formato .doc, .docx (Word) y .pdf.
- Comprueba que el código y el nombre de la asignatura corresponden a la asignatura de la que te has matriculado.
- Tiempo total: **1 hora** Valor de cada pregunta:
- ¿Puede consultarse algún material durante la prueba de síntesis? ¿Qué materiales están permitidos?
- ¿Puede utilizarse calculadora? ¿De qué tipo?
- Si hay preguntas tipo test, ¿descuentan las respuestas erróneas? ¿Cuánto?
- Indicaciones específicas para la realización de esta prueba de síntesis:

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30

Enunciados

Pregunta 1 [50%]

Continuando la aplicación para la gestión de la pandemia de COVID, en este ejercicio se trabajará solo con los siguientes tipos:

```
const
    MAX_PERSONS: integer = 100;
    MAX_APPOINTMENTS: integer = 20;
end const

type
    tDate = record
        day : integer;
        month : integer;
        year : integer;
    end record

    tTime = record
        hour : integer;
        minutes : integer;
    end record

    tDateTime = record
        date : tDate;
        time: tTime;
    end record

    tVaccine = record
        name : string;
        required : integer;
        days : integer;
    end record

    tPerson = record
        document : string;
        name : string;
        surname : string;
        email : string;
        address : string;
        cp : string;
        birthday : tDate;
    end record

    tPopulation = record
        elems : vector[MAX_PERSONS] of tPerson;
        count : integer;
    end record
```

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30

```

tAppointment = record
    timestamp : tDateTime;
    person : tPerson;
    vaccine : tVaccine;
end record

tAppointmentData = record
    elems : vector[MAX_APPOINTMENTS] of tAppointment;
    count : integer;
end record
end type

```

Dada la función:

```

function population_with_vaccine(appointments: tAppointmentData, vaccine: string):
    tPopulation

```

que dadas unas citas y una vacuna devuelve la lista de personas que tienen una cita programada para esa vacuna. Para simplificar el ejercicio podemos asumir que la misma persona tendrá como máximo una cita, es decir, una persona no podrá tener dos o más citas programadas.

Se pide:

- a) **[10%]** Indica las pre-condiciones para la función *population_with_vaccine*. En este caso, se asume que la lista de citas no está vacía y que existe por lo menos alguna persona con cita para la vacuna que nos dan. **Justifica la respuesta con una o dos líneas de texto.**

$\{ \text{appointments} = \text{APPOINTMENTS} \text{ and } \text{APPOINTMENTS.count} > 0 \text{ and } \exists i : 0 < i \leq \text{APPOINTMENTS.count} : \text{APPOINTMENTS.elems}[i].\text{vaccine.name} = \text{vaccine} \}$

La lista de citas no está vacía si tenemos por lo menos un elemento. Nos dice el enunciado que existe una cita con la vacuna, no hará falta comprobar a qué persona está asociada

- b) **[40%]** Implementa la función en **lenguaje algorítmico** aplicando la metodología de diseño descendente. También es necesario implementar todas las funciones que necesites crear en los diferentes niveles. Para cada método, describe brevemente la función que realizan (escribe una o dos líneas de texto por método). **No se valorarán métodos que no tengan su correspondiente descripción.**

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30

Nivel 1

```

function population_with_vaccine( appointments: tAppointmentData, vaccine: string):
tPopulation
var
    population: tPopulation;
    i: integer;
end var
    { Initialize variables }
    init_population(population);

    { Loop all the appointments }
    for i:=1 to appointments.count do
        { Filter by vaccine }
        if person_uses_vaccine(appointments.elems[i], vaccine) then
            { Add person to population with searched vaccine }
            add_person(population, appointments.elems[i]);
        end if
    end for

    return population;
end function

```

Nivel 2

```

    { Check if the person uses a vaccine on the appointment }
function person_uses_vaccine( appointment: tAppointment, vaccine: string): boolean
    return appointment.vaccine.name = vaccine;
end function

    { Initialize the population table }
action init_population(inout population: tPopulation)
    population.count := 0;
end action

    { Add a person to the population table }
    { We assume that it is not necessary to check "duplicate" person }
action add_person(inout population: tPopulation, in appointment: tAppointment)
    { Update length of the population }
    population.count := population.count + 1;

    { Add person to the population }
    { It also possible to make a copy, not just point to the same person }
    population.elems[population.count] := appointment.person;
end action

```

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30

Pregunta 2 [50%]

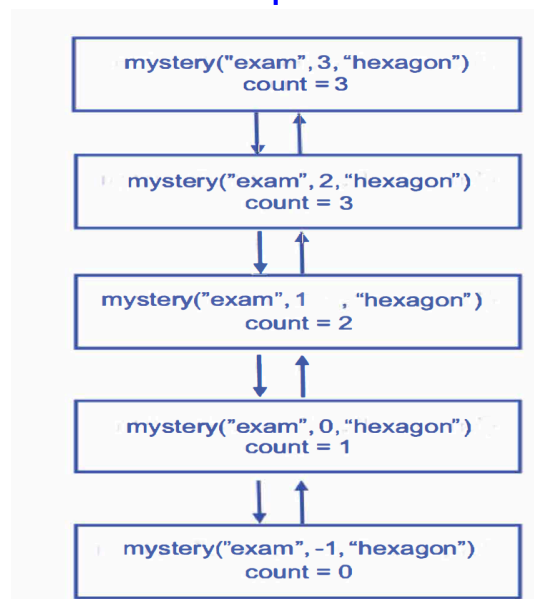
Dada la siguiente implementación en lenguaje C de un método *mystery*, donde la función:
int search(char a, char c[N])
 busca el carácter *a* en la cadena *c*, retornando 1 si lo encuentra, 0 en caso contrario:

```
#define N 128
int mystery (char p[N], int x, char q[N])
{
    assert (-1 <= x && x < N);
    int count;
    if (x < 0)                                     (1)
        count = 0;                                (2)
    else                                           (3)
        count = mystery (p, x-1, q) + search(p[x], q); (4)
    return count;                                 (5)
}
```

- a) [10%] Explica qué hace este método *mystery*, es decir, qué función realiza. También muestra paso a paso (incluyendo un diagrama de cajas para secuencia de llamadas y sus parámetros) el funcionamiento de *mystery* para la siguiente invocación:

```
int ret;
ret = mystery("exam", 3, "hexagon");
```

Este método devuelve el número de caracteres de los primeros *x+1* caracteres de la cadena *p* que están presentes en la cadena *q*.



Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30

b) [5%] Explica qué consecuencias tendría sustituir la línea (4) por esta línea:

count = mystery (p, x, q) + search(p[x], q);

Si el valor del parámetro **x** es inicialmente mayor que -1, entonces nunca acabará la recursión ya que la variable **x** nunca cambia de valor en llamadas recursivas sucesivas.

c) [20%] Transforma este método recursivo en un método iterativo en lenguaje C. Describe brevemente la implementación adoptada (en caso contrario, no se evaluará el ejercicio).

```
#define N 128
int mystery (char p[N], int x, char q[N])
{
    assert (-1 <= x && x < N);
    count = 0;
    while (x >= 0) {
        count = count + search(p[x], q);
        x = x-1;
    }
    return count;
}
```

El método recorre en cada iteración del bucle una posición del vector **p** (comenzando en la posición **x**) analizando a través de la función **search** si este elemento se encuentra en el vector **q**. En cada iteración se decrementa **x** en una posición. El bucle finaliza cuando la posición **x** alcanza el valor 0.

d) [15%] Analiza la complejidad de este método (en su versión recursiva) asumiendo que la función de tiempo de **search(a, c)** es $T_{\text{search}} = s$, donde **s** es la longitud de **c** en caracteres.

Para analizar la complejidad computacional de esta función, hay que observar que a cada llamada recursiva el valor del parámetro **x** se decrementa en una unidad hasta que llega al valor -1 donde finaliza la recursividad. Por lo tanto, la función de tiempo se define en función del parámetro **x**.

El caso base (líneas 1 y 2) se limita a un condicional y una asignación. Al tratarse de dos operaciones elementales se agrupa su coste en una única constante k_1 .

Para calcular el caso recursivo, partimos de un condicional (línea 1), una asignación, una operación aritmética y un acceso a vector (línea 4) con costes constantes agrupados en k_2 , a los que tenemos que sumar el coste de la llamada recursiva y de la llamada a la función **search**. Para el cálculo de la llamada recursiva tendremos en cuenta el peor caso posible, que es que la cadena **q** tenga siempre **N** caracteres, por lo que el tiempo será $T_{\text{search}} = N$

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30

Podemos definir la función de tiempo de la siguiente manera:

$$\begin{aligned} T(x) &= k_1, & \text{si } x = -1 \\ T(x) &= k_2 + T(x-1) + N, & \text{si } x \geq 0 \end{aligned}$$

Se resuelve la recurrencia repitiendo la sustitución recursiva un total de i veces:

$$\begin{aligned} T(x) &= k_2 + T(x-1) + N = \\ &= k_2 + (k_2 + T((x-1)-1) + N) + N = 2 \cdot k_2 + T(x-2) + 2 \cdot N = \\ &= 2 \cdot k_2 + (k_2 + T((x-2)-1) + N) + 2 \cdot N = 3 \cdot k_2 + T(x-3) + 3 \cdot N = \\ &= 3 \cdot k_2 + (k_2 + T((x-3)-1) + N) + 3 \cdot N = 4 \cdot k_2 + T(x-4) + 4 \cdot N = \\ &= \dots = \\ &= i \cdot k_2 + T(x-i) + i \cdot N \end{aligned}$$

Calculamos el valor de la variable i que hace que se pueda usar la parte no recurrente de la definición de $T(x)$:

$$\begin{aligned} x - i &= -1 \\ x + 1 &= i \end{aligned}$$

Finalizamos el cálculo de $T(x)$ con el valor calculado por la variable i :

$$\begin{aligned} T(x) &= i \cdot k_2 + T(x-i) + i \cdot N = (x+1) \cdot k_2 + T(x-x-1) + (x+1) \cdot N = \\ &= x \cdot k_2 + k_2 + T(-1) + x \cdot N + N = x \cdot k_2 + k_2 + k_1 + x \cdot N + N = \\ &= x \cdot (k_2 + N) + k_2 + k_1 + N \end{aligned}$$

La conclusión es que la función tiene una complejidad lineal **O(x)**.

Prueba de síntesis 2021/22-2

Asignatura	Código	Fecha	Hora inicio
Prácticas de programación	75.555	18/6/2022	17:30