Fundamentos de Programación Examen Final – 15/02/2018

Observaciones generales:
- Realizar los ejercicios en hojas separadas. Tiempo para Regulares: 2 hs. min.
- LEER antes de resolver y consultar cualquier duda. Tiempo para los Libres: 2 hs. 15 min.
- Para aprobar el examen hay que tener un por lo menos el 60% de la práctica y por lo menos el 60% de la teoría.

Ejercicio 1 (Regulares: 40 puntos - Libres: 25 puntos)

En un taller mecánico trabajan diferentes especialistas y cada uno de ellos cobra un valor por hora de mano de obra ya pautado.
Al inicio se ingresa la información de los 10 especialistas que trabajan en el taller: código de especialista (1 a 10), apellido y nombre y valor a cobrar por hora.
Al final del mes se ingresan las reparaciones producidas en el taller, ingresando para ello: patente del vehículo (alfanumérico), código de especialista (de 1 a 10), cantidad de horas que llevó la realización del trabajo. Una patente igual a “99 ZZZ 99” indica el fin de datos.
Se desea:
a.) Realizar un listado de los especialistas ordenado por apellido y nombre con la siguiente información: código de especialista, apellido y nombre, total de horas trabajadas en el mes, cantidad de veces que se requirió su labor, total a pagar.
b) Informar que especialista trabajó la mayor cantidad de horas.
c) Calcular e informar cuantas veces la patente “AB 123 AA” ingresó en reparación y el promedio de horas que se utilizó en el vehículo.
OBSERVACIÓN: Dos de los ítems a), b) y c) deben ser resueltos utilizando funciones. Escoja dos de ellos.

Ejercicio 2 (Regulares: 40 puntos - Libres: 25 puntos)

Una clínica veterinaria para perros ofrece 3 servicios: peluquería, control sanitario y vacunación.
La peluquería cuesta $350 y el control sanitario $450. Las 7 vacunas que ofrece la clínica son: Parvovirus Canino (1), Moquillo Canino (2), Hepatitis Infecciosa Canina (3), Leptospirosis (4), Parainfluenza (5), Infección por Coronavirus (6) y Antirrábica (7).
Los precios de las vacunas se ingresan al principio en orden de acuerdo a su código.
A continuación se ingresa el código de los N clientes (desde 1 hasta N) y el nombre de cada una de las mascotas que atiende la clínica. El valor N se ingresa al inicio.
Luego se ingresan cada una de las atenciones de la clínica durante el mes. Por cada una de ellas se ingresa: código del cliente, día de atención (1 a 7), servicio ('P': peluquería, 'C': control, 'V': vacunación). Además, si el servicio solicitado es vacunación se debe ingresar el código de la vacuna.
Los datos se ingresan ordenados por día de atención y puede existir más de una visita en el mes para el mismo cliente. Un código de cliente igual a 0 indica el fin de datos.
Se pide informar:
a) Listado del total adeudado por cada cliente.

Código de cliente	Nombre de la mascota	Total adeudado
99999	XXXXXXXXXXXXX	999999

b) Cantidad de vacunas vendidas de cada tipo.
c) Generar un archivo llamado **`antirrabica.txt`**, con los siguientes datos: código de cliente y nombre de la mascota, para aquellas mascotas a las que se les aplicó la vacuna código 7.

Ejercicio 3 (Regulares: 0 puntos - Libres: 25 puntos)

Implementar en C++ la siguiente función: **`bool agregar (int a[], int& tamanio, int valor)`**;
La misma recibe como parámetros de entrada un arreglo “a” de enteros, la cantidad de elementos que contiene en la variable “tamanio” y un valor entero a ser agregado.
Esta función verifica si "valor" está en el arreglo a; en caso de encontrarlo no realiza modificación alguna, pero si no lo encuentra, lo agrega al final. Como esto implica una modificación del tamaño del arreglo debe actualizarse además el parámetro “tamanio” al valor correspondiente.
El retorno de la función es true si se produjo alguna modificación en el vector o false en caso contrario.

Ejemplo:
Antes de llamar a la función

a[] = {1, 5, 9, 1, 10, 14, 14, 14, 1, 22, 9, 3};
t = 12;

Luego de la llamada a la función “retorno = agregar (a, t, 99);”

a[] = {1, 5, 9, 1, 10, 14, 14, 14, 1, 22, 9, 3, 99};
t = 13;
retorno = true;

Fundamentos de Programación Examen Final – 05/07/2018

Observaciones generales:
- Realizar los ejercicios en hojas separadas. Tiempo para Regulares: 2 hs. min.
- LEER antes de resolver y consultar cualquier duda. Tiempo para los Libres: 2 hs. 15 min.

Ejercicio 1 **(Regulares: 40 puntos - Libres: 25 puntos)**

Una institución educativa desea realizar un seguimiento experimental sobre 130 participantes previamente seleccionados. El seguimiento consta de 7 pruebas físico-intelectuales, codificadas de 1 a 7 y calificadas de 0 a 10.
Cada participante posee un código de 1 a 130. Al comienzo, se ingresa el DNI y el nombre y apellido de cada participante ordenado por su código.
Los alumnos pueden optar por no realizar alguna prueba, con lo cual obtendrán un -1 en la misma.
Se conocen por cada prueba realizada por un participante: Código de participante (1 a 130), Código de la prueba (1 a 7) y puntaje obtenido. Los datos finalizan con código de participante = 99. Los datos se ingresan sin ningún orden.
Se desea:
a) Realizar un listado ordenado en forma decreciente por puntaje de cada participante con los siguientes datos:

Nombre y apellido	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**Puntaje**
xxxxxxxxx	**Xx**	**Xx**	**Xx**	**Xx**	**Xx**	**Xx**	**Xx**	**Xxx**

OBS 1: El puntaje es el promedio de notas obtenidas en las 7 pruebas.
OBS 2: El ordenamiento debe ser efectuado por una función llamada ORDENA (determine el tipo y los parámetros que considere necesarios). Esta función ***no debe efectuar el listado***, el mismo debe efectuarse en el main.
b) Generar un archivo **`sin_rendir.txt`** → Donde se guardarán los siguientes datos: **`DNI, Nombre y apellido, código de prueba`**, de aquellos alumnos que posean un valor -1 en una o más pruebas. Podrá haber DNI repetidos, dado que se guarda una línea por cada prueba no efectuada.

Ejercicio 2 **(Regulares: 40 puntos - Libres: 25 puntos)**

El gobierno nacional desea realizar un control sobre la producción de soja en diferentes localidades del país previamente seleccionadas. Las localidades se encuentran codificadas con un código alfanumérico (del tipo 01PAR3100).
Al principio se ingresa por cada localidad: código de localidad, nombre de la localidad y cantidad de establecimientos que cosecharon soja. Estos datos se ingresan sin ningún orden y el fin de datos está dado por código de localidad = “0x”.
Luego se conocen los datos correspondientes a las toneladas de soja obtenidas por cada establecimiento en cada localidad, sin orden alguno: trimestre del año (1, 2, 3 ó 4), código de localidad y cantidad de toneladas de soja. El fin de datos está dado por trimestre = 9 y puede venir más de una carga por localidad y trimestre.
Se desea:
a) Obtener un listado como el siguiente:

CÓDIGO DE LOCALIDAD	**TOTAL LT.1º TRIM.**	**TOTAL LT.2º TRIM.**	**TOTAL LT.3º TRIM.**	**TOTAL LT.4º TRIM.**
Xxx	**xxxx**	**xxxx**	**xxxx**	**xxxx**
Xxx	**xxxx**	**xxxx**	**xxxx**	**xxxx**

b) Ingresar un nombre de Localidad y determinar si existe. Si existe, calcular el promedio anual de toneladas de soja de esa localidad. Si no existe, mostrar el siguiente mensaje: “Localidad no encontrada”. (OBS: El promedio anual se obtiene dividiendo el total de toneladas de soja de cada localidad, por el total de establecimientos de la misma).

Ejercicio 3 **(Regulares: 0 puntos - Libres: 20 puntos)**

Implementar en C++ la siguiente función: bool agregarordenado (int a[ ], int &tamanio, int valor);
La misma recibe como parámetros de entrada un arreglo “a” de enteros, la cantidad de elementos que contiene en la variable “tamanio” y un valor entero a ser agregado.
Esta función verifica si "valor" está en el arreglo a; en caso de encontrarlo no realiza modificación alguna, pero si no lo encuentra, lo agrega en la posición que le corresponde para mantener el orden creciente de los datos del arreglo. Como esto implica una modificación del tamaño del arreglo debe actualizarse además el parámetro “tamanio” al valor correspondiente.
El retorno de la función es true si se produjo alguna modificación en el vector o false en caso contrario. Ejemplo:
Antes de llamar a la función

a[ ] = {1, 5, 9, 11, 12, 14, 15, 18, 21, 22, 29, 35};
t = 12;

Luego de la llamada a la función “retorno = agregarordenado (a, t, 32);” :

a[ ] = {1, 5, 9, 11, 12, 14, 15, 18, 21, 22, 29, 32, 35};
t = 13;
retorno = true;

Fundamentos de Programación Examen Final – 04/10/2018

Observaciones generales:
- Realizar los ejercicios en hojas separadas.
- LEER antes de resolver y consultar cualquier duda.

Tiempo para Regulares: 2 hs. min.
Tiempo para los Libres: 2 hs. 15 min.

Ejercicio 1 (Regulares: 40 puntos - Libres: 25 puntos)

La Fundación Bunge & Born otorga becas agrotécnicas para contribuir a que más chicos de escuelas primarias rurales finalicen sus estudios.
Se otorga un monto anual distribuido en cuotas mensuales para que alumnos, de entre trece y diecisiete años, puedan cubrir gastos de material, transporte, alimento y hospedaje (en caso que la escuela sea también albergue).
Para ello, lo primero que se ingresa es la cantidad de becas a otorgar y el monto anual a asignar por beca para el año 2018.
De la convocatoria se ingresa fecha de inicio, fecha de fin de la convocatoria, promedio requerido, y se sabe que hubo 400 postulantes de todo el país.
De cada postulante se ingresa: documento, apellido y nombre, edad, promedio, nombre de la escuela, provincia y fecha de presentación.
Se desea:
1) Informar cuántos postulantes quedaron fuera de la convocatoria por presentarse luego de la fecha de finalización de la convocatoria.
2) Informar el siguiente detalle, de la cantidad de becas otorgadas, ordenado de manera descendente por promedio.

Apellido y nombre	Nombre Escuela	Edad	Provincia	Promedio
Xxxxxxxxxx	Xxxxxxxxxx	Xx	Xxxxxxx	Xx
Xxxxxxxxxx	Xxxxxxxxxx	Xx	Xxxxxxx	Xx
……				

3) Informar el monto total de la provincia que obtuvo más becas.
4) Informar cantidad de inscriptos por edad.

Ejercicio 2 (Regulares: 40 puntos - Libres: 25 puntos)

Una empresa metalúrgica distribuye sus productos a sus 5 centros (codificados de 0 a 4) que tiene en el país y desea realizar el control de las entregas. Para ello, son ingresados primero los datos de los 200 productos que comercializa: código producto (0 a 199), cantidad mínima de stock y cantidad disponible.
A continuación se ingresan los datos de los pedidos solicitados el día anterior de los distintos productos desde los diferentes centros. Se ingresan sucesivamente: código de centro, código de producto, cantidad de solicitada. Estos datos se ingresan sin orden. Un código 999 determina el fin de la carga de datos.
A medida que se ingresan los pedidos se debe verificar si existe disponibilidad en el stock para cumplir con la cantidad solicitada del producto. En caso que sea suficiente, se genera el mismo y se debe actualizar la cantidad disponible.
Cuando la cantidad disponible no es suficiente ese pedido debe quedar como pendiente de entrega.

Se desea:
1) Crear una función COMPROBAR_STOCK, que verifique si es posible o no cumplir con la solicitud.
2) Generar un archivo con los pedidos PENDIENTES de entrega, con los siguientes datos: Número del Centro Distribución, Código Pedido, Cantidad pedida. Este archivo debe tener sus datos ordenados por Número de centro de distribución.
3) Generar un informe de todos los productos que han tenido movimiento

Código de producto	Cantidad mínima de stock	Cantidad disponible	Cantidad Entregada
Xxx	xxx	xxx	xxx

Ejercicio 3 (Regulares: 0 puntos - Libres: 20 puntos)

Considere la siguiente definición de estructura:

```
struct riesgo_cardiaco {
      string paciente;
      int  ritmo_card,
      int presion,
      int edad;
      float altura,
```

```
  float peso};
```

Se pide: Escriba una función C++ llamada `mayor_riesgo(..)` de tipo **riesgo_cardíaco** que devuelva los datos de la persona con mayores probabilidades de sufrir un ataque al corazón. La función recibe como parámetros un arreglo de tipo riesgo_cardiaco con los datos de varias personas y un entero n con la longitud del arreglo (cantidad de personas a analizar). La función determina la persona de mayor riesgo cardíaco a través de una índice que se calcula:
indice _riesgo =4*presion+ritmo_card+ peso/(3*altura)+edad/3; cuanto mayor es este índice, mayor es el riesgo de sufrir un ataque cardíaco.
En lo referente a la función main, realice las acciones necesarias para llamar a la función y para recibir el resultado de la misma.

Fundamentos de Programación Examen Final – 06/12/2018

Observaciones generales:
- Realizar los ejercicios en hojas separadas.
- LEER antes de resolver y consultar cualquier duda.

Tiempo para Regulares: 2 hs. min.
Tiempo para los Libres: 2 hs. 15 min.

Ejercicio 1 (Regulares: 40 puntos - Libres: 25 puntos)

El INDEC cuenta con un archivo <indec2017.txt> donde se detallan los índices de inflación por provincias y por mes registrados durante el 2017. La información de cada registro consiste en: Cod. Provincia (1 a 24), mes (1 a 12) e índice de inflación (los datos se encuentran separados por espacios). Estos datos no tienen orden.
Luego se ingresa la información de productos de la canasta básica de alimentos, informados por las distintas provincias, ingresando por cada uno: mes de relevamiento (1 a 12), descripción del producto, valor del producto y código de provincia. Un mes de relevamiento igual a 0 indica el fin de datos.
Se desea:
1) Generar un archivo <productos2017.txt> que contenga la siguiente información: descripción del producto y valor a diciembre del 2017. El valor a diciembre del 2017 se determinará incrementando el valor del producto de acuerdo al índice de inflación para la provincia desde el mes de relevamiento hasta diciembre*.
2) Informar por cada provincia cantidad de productos relevados.
3) Indicar a que provincia y mes corresponde el mayor índice de inflación. Implemente para este punto, una función que devuelva la información solicitada.
* Implemente una función que devuelva el porcentaje de aumento a aplicar.

Ejercicio 2 (Regulares: 40 puntos - Libres: 25 puntos)

El municipio ha implementado un sistema de estacionamiento medido diariamente de 8.00 a 13.00 hs, y por el servicio, percibe el 20% de lo recaudado por cada ticket de estacionamiento. Al principio se ingresa el precio unitario por hora del estacionamiento.
Se cuenta con un archivo <personal_mes.txt> donde se encuentran los datos de los 90 agentes habilitados por el municipio. Cada registro contiene: Código del personal (alfanumérico), Apellido y Nombre, Dni, Dirección y teléfono.
Cada vez que alguien estaciona, se genera un ticket con los siguientes datos: código del personal habilitado (alfanumérico), Nro de comprobante (alfanumérico), Patente (alfanumérico), hora de inicio del estacionamiento (solo hora), cantidad de horas que estará estacionado (una, dos, etc.), monto que el usuario deberá pagar, fecha (ddmmaaaa). El fin está dado por código del personal habilitado =”x”.
Se desea:
1) Generar una estructura que me permita guardar los datos de cada ticket.
2) Emitir el siguiente listado ordenado por código de personal habilitado:

Código de Personal Nro de Crompobante Monto del comprobante Para Municipio
xxxx xxxxx 999,99 99,99
Cantidad total recaudada 99999,99 Monto total a devolver al municipio: 9999,99

3) Informar Código del personal (alfanumérico), Apellido y Nombre, Dni, del empleado que recaudo el mayor monto.

Ejercicio 3 (Regulares: 0 puntos - Libres: 20 puntos)

Se desea contar con una aplicación que realice ciertas acciones sobre un texto que se ingresa en una variable string. La aplicación debe pasar todas las palabras de más de 3 caracteres a un arreglo. En el arreglo no puede haber palabras repetidas y deberán contarse la cantidad de repeticiones de cada palabra.
Luego se deben ordenar las palabras de acuerdo a su largo, colocando la de mayor longitud al inicio.
Informar la lista de palabras con sus respectivas repeticiones.
Utilizar al menos 2 funciones.

Fundamentos de Programación Examen Final – 20/12/2018

Observaciones generales:
- Realizar los ejercicios en hojas separadas.
- LEER antes de resolver y consultar cualquier duda.

Tiempo para Regulares: 2 hs. min.
Tiempo para los Libres: 2 hs. 15 min.

Ejercicio 1 (Regulares: 40 puntos - Libres: 25 puntos)

En una Ruta Nacional hay una Estación de Peajes que cuenta con 5 cabinas de cobro.
Los vehículos a los cuales se les cobra peajes están codificados de la siguiente manera: 1 – Motos, 2 – Autos, 3 – Camiones, 4 – Colectivos, 5 – Otros.
Se ingresa al principio el precio del peaje de cada tipo de vehículo.
Cada vez que pasa un vehículo se registra: hora de paso (00, 01, 02, ...,23), tipo de vehículo (1 a 5), Patente (es un dato del tipo string, que puede tener básicamente dos formas: xxx999 o xx999xx, donde x son letras y 9 son números) y cabina (1 a 5). (obs: se consideran horas sin minutos). El fin de datos está dado por hora = 25.
Durante las ”Horas Pico” (08 a 12 y 16 a 20) el precio del peaje es un 20% más caro.
1) Se desea conocer la siguiente información:
a- Cantidad de vehículos que pasaron por cada cabina.
b- Cuántos autos cruzaron en “Hora Pico” por la estación de peajes, discriminado por cada cabina.
c- Cuantos camiones cruzaron durante el día por la estación de peajes.
d- El total recaudado durante el día por cada cabina y el total general.
e- Cantidad de autos con patentes “nuevas” (formato xx999xx).
2) Generar un archivo llamado MONTOS que contenga la siguiente información de cada cabina: Número de cabina | Total recaudado en la misma.

Ejercicio 2 (Regulares: 40 puntos - Libres: 25 puntos)

Una empresa multinacional, cuenta con varios departamentos, entre ellos **Exportación**, encargado de enviar los productos a diferentes países.
Como primer dato, se ingresa la fecha actual: Día (1 a 31), Mes (1 a 12) y Año.
Fabrican 50 productos diferentes (codificados de 1 a 50) y son enviados a 20 países (codificados de 1 a 20). Se conocen los costos unitarios de envío de cada producto para cada país, sin orden alguno, ingresando: Código de Producto, Código de País y Costo de envío.
Luego se ingresan los pedidos semanales de la siguiente manera: Código de Producto, Código de País, Cantidad a enviar, Mes (1 a 12) y Año. Estos datos se cargan sin orden y un Cód. De Producto = 0 indica el fin de la carga.
Sólo se enviarán los pedidos que cumplan con la condición de que el mes sea igual o superior al mes actual y el año sea el vigente. Si se trata de un año superior, deberá generarse un archivo <pendientes.txt> que almacene todos los datos leídos.
Al finalizar la carga, se pide mostrar dos informes:
a) Cantidad total de productos a enviar, discriminado por producto y por país.
b) Total a cobrar por envío, discriminado por producto y por país.
Ambos informes, deberán tener el siguiente formato de encabezado:

Producto/País 1 2 3 4 5....20
1
2
…
50

El main, deberá tener el siguiente formato:

```
int main ( )
{
//agregar definiciones necesarias
cout<<”Ingreso de fecha: “;
cin>>dia_a; cin>>mes_a; cin>>anio_a;
CargaPU(.....);
CargaPedidos(.....);
GenerarPendientes(.....);
Muestra_Envíos(.....);
Muestra_Cobrar(.....);
return 0;
}
```

OBSERVACIONES:

- Los parámetros de las funciones deberán ser fijados por Ud.

- Todas las funciones son de tipo void.

- Deberá agregar los prototipos de las funciones en el lugar correspondiente.

- Puede generar otras funciones, pero no deben ser convocadas desde el main.

Ejercicio 3 **(Regulares: 0 puntos - Libres: 20 puntos)**

Implementar en C++ la siguiente función: **bool eliminar (int a[], int& tamanio, int valorEliminar);**

La misma recibe como parámetros de entrada un arreglo “a” de enteros, la cantidad de elementos que contiene en la variable “tamanio” y un valor entero a ser eliminado.
Esta función procesa el arreglo eliminando los elementos iguales al parámetro “valorEliminar”. Como esto implica la reducción del tamaño del arreglo debe actualizarse además el parámetro “tamanio” al valor correspondiente.
El retorno de la función es true si se produjo alguna modificación en el vector o false en caso contrario (o sea, cuando “valorEliminar” no está en el arreglo). Ejemplo.
Antes de llamar a la función:

```
a[] = {1, 5, 9, 1, 10, 14, 14, 14, 1, 22, 9, 3};
t = 12;
bool retorno = false;
```

Luego de la llamada a la función “retorno = eliminar (a, t, 1);”

```
a[] = {5, 9, 10, 14, 14, 14, 22, 9, 3};
t = 9;                  // 9 es el valor de t
retorno = true;         //true es el valor de retorno
```

Fundamentos de Programación Examen Final – 01/03/2018

Observaciones generales:
- Realizar los ejercicios en hojas separadas. Tiempo para Regulares: 2 hs. min.
- LEER antes de resolver y consultar cualquier duda. Tiempo para los Libres: 2 hs. 15 min.
- Para aprobar el examen hay que tener un por lo menos el 60% de la práctica y por lo menos el 60% de la teoría.

Ejercicio 1 (Regulares: 35 puntos - Libres: 25 puntos)

Una entidad de salud desea realizar un estudio estadístico referido a 3 enfermedades: 1: Hipertensión, 2: Diabetes, 3: Colesterol. Para ello, desea contar con un registro de la cantidad de personas que padecen esas enfermedades y son atendidas en diferentes centros de salud de la provincia, durante un semestre.
Al principio se conocen los datos de los 50 centros de salud: código de centro de salud (1-50), nombre y dirección.
A continuación, se ingresan los datos de los pacientes atendidos en el período: código de centro de salud (1-50), código de enfermedad (1-3), número de mes y cantidad de pacientes atendidos. Estos datos no vienen ordenados y puede haber varios datos para un mismo centro de salud. La carga finaliza con un código de centro de salud igual a 99.
Se pide
a) Generar 3 archivos: `hiperten.txt`, `diabetes.txt`, `colest.txt`, con la siguiente información: Código Del Centro De Salud | Nombre Del Centro De Salud | Cantidad De Pacientes atendidos en el semestre.
Obs.: Cada archivo deberá contener sólo los datos de la enfermedad a la que hace referencia su nombre.
b) Buscar e informar si el centro de salud “ARTURO OÑATIVIA” ha atendido más de 140 personas hipertensas.
c) Informar Nombre del Centro de salud que atendió la mayor cantidad de pacientes con Diabetes. Utilice una función para resolver este ítem.

Ejercicio 2 (Regulares: 35 puntos - Libres: 25 puntos)

Una Compañía de Seguros desea calcular el monto que debe pagar a cada uno de sus promotores en función de las comisiones por los seguros contratados y de la bonificación.
Tiene 25 promotores- codificados de 1 a 25- encargados de realizar seguros de distintas líneas.
La compañía otorga a cada promotor una bonificación que corresponde a las ganancias del mes anterior. Para ello, ingresa como primeros datos, el porcentaje y luego el monto de ganancias.
El monto a distribuir se calcula de la siguiente manera: MONTO A DISTRIBUIR= GANANCIAS * PORCENTAJE/100
Además se conoce de cada operación realizada por cada promotor los siguientes datos: código de promotor (1 a 25), línea de seguro contratada por el cliente (1: automotor, 2: vivienda, 3: vida, 4: otros), monto del seguro, monto de la comisión que le corresponde. Estos datos NO vienen ordenados y pueden venir más de un juego de datos para cada promotor. El fin de datos se produce con un código de promotor igual a 999.
Se desea:
a) Calcular e informar el total que le corresponde cobrar a cada promotor (suma de comisiones, más bonificación del mes anterior).
b) Calcular e informar el total a pagar por la compañía en todo concepto.
c) Emitir un listado ordenado por Monto total de todos los seguros, discriminado por promotor y por tipo de seguro, con el siguiente formato:

```
Cod.Promotor  Monto total  Monto Automotor  Monto Vivienda  Monto Vida  Monto otros
   99           99999,99     99999,99         99999,99        99999,99    99999,99
```

Ejercicio 3 (Regulares: 0 puntos - Libres: 20 puntos)

Realizar una función recursiva que obtenga el producto de dos números enteros.
Antes de llamar a la función recursiva, deberá corroborar que los números ingresados sean positivos y mayores a 0.