02 – 03 - 2013	Cantidad de Hojas:___	Nota:_____ Evaluó Prof:_____	
Apellido y Nombre:_____		Legajo:_____	Curso con:_____

ALGORITMOS Y ESTRUCTURA DE DATOS
EXAMEN FINAL

La Cámara de Empresarios del Software y Sistemas de Información CESSI, lanza el concurso 2013 de los Premios Sadosky a la inteligencia Argentina, con los que destaca a profesionales, industrias y universidades del sector del Software y necesita realizar la lista de los nominados. Para ello cuenta con los siguientes archivos de registros:

RUBROS.dat (sin orden, con 1 registro por c/u de las 3 categorías de los 15 rubros existentes)

rubro	categoría
15 caracteres	15 caracteres

POSTULANTES.dat (ordenado por rubro y categoría)

			postulante	
rubro	categoría	proyecto	id	nombre
		50 caracteres	4 dígitos	50 caracteres

VOTOS.dat (sin orden, con 1 registro por c/u de los votos de los asociados)

asociado	rubro	categoría	idPostuante
50 caracteres			

Se pide desarrollar un programa que:

1. Genere el siguiente archivo con datos de los nominados a los premios, que serán aquellos postulantes que hayan obtenido las 3 mayores cantidades de votos por cada rubro/categoría.

NOMINADOS.dat (ordenado por rubro, categoría y cantidad de votos)

rubro	categoría	nombrePostulante	proyecto	cantidadVotos
				word

2. Al finalizar el proceso imprima un listado informando la cantidad de postulantes y nominados de cada categoría de cada rubro, con el siguiente diseño.

Rubro: XXXXXXXXXX

Categoría	*Postulantes*	*Nominados*
XXXXXXXXXX	99999	99999
XXXXXXXXXX	99999	99999
XXXXXXXXXX	99999	99999

Recursos y restricciones

- Memoria para *arrays*: máximo 2000 bytes.
- Memoria dinámica: nodos de no más de 12 bytes.
- Accesos a disco: 1 acceso secuencial a cada registro de cada archivo más 1 acceso directo al archivo **POSTULANTES.dat**.

NOTA: Se considerará especialmente la adecuada aplicación de la metodología ***top-down***.

02 – 03 - 2013	Cantidad de Hojas:___	Nota:_____ Evaluó Prof:_____	
Apellido y Nombre:_____		Legajo:_____	Curso con:_____

ALGORITMOS Y ESTRUCTURA DE DATOS
EXAMEN FINAL

La Cámara de Empresarios del Software y Sistemas de Información CESSI, lanza el concurso 2013 de los Premios Sadosky a la inteligencia Argentina, con los que destaca a profesionales, industrias y universidades del sector del Software y necesita realizar la lista de los nominados. Para ello cuenta con los siguientes archivos de registros:

RUBROS.dat (con 1 registro por c/u de las 3 categorías de los 15 rubros existentes, sin orden)

rubro	categoria
15 caracteres	15 caracteres

POSTULANTES.dat (ordenado por `postulante.id`)

			postulante	
rubro	categoria	proyecto	id	nombre
		50 caracteres	4 dígitos	50 caracteres

VOTOS.dat (ordenado por `asociado`, con 1 registro por c/u de los votos de los asociados)

asociado	rubro	categoria	idPostuante
50 caracteres			

Se dispone de la función `toId` que, dado un rubro y una categoría retorna un identificador numérico único de tipo `word` con la siguiente característica: sean r_1, r_2 dos rubros y c_1, c_2 dos categorías entonces si la combinación $r_1+c_1 >= r_2+c_2$ se verifica que:
`toId(`r_1`,`c_1`) >= toId(`r_2`,`c_2`)`

Se pide desarrollar un programa que:

1. Genere el siguiente archivo (ordenado por rubro/categoría) con datos de los nominados a los premios, que serán aquellos postulantes que hayan obtenido las 3 mayores cantidades de votos por cada rubro/categoría.

NOMINADOS.dat

rubro	categoria	nombrePostulante	proyecto	cantidadVotos
				word

2. Imprima un listado informando por por cada rubro, todas las categorías que quedaron vacantes por falta de postulantes, con el siguiente diseño.

Rubro: XXXXXXXXXX
    Categorías Vacantes
        XXXXXXXXXX
        XXXXXXXXXX
        XXXXXXXXXX

Recursos y restricciones

- Memoria para *arrays*: 0 bytes.
- Memoria dinámica: nodos de no más de 12 bytes.
- Accesos a disco: 1 acceso secuencial a cada registro de cada archivo más 1 acceso directo al archivo **POSTULANTES.dat** y 2 accesos directos al archivo **RUBROS.dat**.

NOTA: Se considerará especialmente la adecuada aplicación de la metodología ***top-down***.

02 – 03 - 2013	Cantidad de Hojas:___	Nota:_____ Evaluó Prof:_____	
Apellido y Nombre:_____		Legajo:_____	Curso con:_____

ALGORITMOS Y ESTRUCTURA DE DATOS
EXAMEN FINAL

La Cámara de Empresarios del Software y Sistemas de Información CESSI, lanza el concurso 2013 de los Premios Sadosky a la inteligencia Argentina, con los que destaca a profesionales, industrias y universidades del sector del Software y necesita realizar la lista de los nominados. Para ello cuenta con los siguientes archivos de registros:

RUBROS.dat (con 1 registro por c/u de las 3 categorías de los 15 rubros existentes, ordenado por rubro y categoría)

rubro	categoría
15 caracteres	15 caracteres

POSTULANTES.dat (ordenado por `postulante.id`)

			postulante	
rubro	categoria	proyecto	id	nombre
		50 caracteres	4 dígitos	50 caracteres

VOTOS.dat (ordenado por `asociado`, con 1 registro por c/u de los votos de los asociados)

asociado	rubro	categoria	idPostuante
50 caracteres			

Se dispone de las siguientes funciones:

- `toRubId` que, dado un rubro retorna un identificador numérico único entre 1 y 15; y
- `toCatId` tal que: `toCatId( toRubId(`*rubro*`)`, *categoria* `)` = *n* donde *n* = [1..3].

Se pide desarrollar un programa que:

1. Genere el siguiente archivo (ordenado por rubro/categoría) con datos de los nominados a los premios, que serán aquellos postulantes que hayan obtenido las 3 mayores cantidades de votos por cada rubro/categoría.

NOMINADOS.dat

rubro	categoría	nombrePostulante	proyecto	cantidadVotos
				word

2. Imprima un listado informando por cada rubro, todas las categorías que quedaron vacantes por falta de postulantes, con el siguiente diseño.

Rubro: XXXXXXXXXX
    Categorías Vacantes
        XXXXXXXXXX
        XXXXXXXXXX
        XXXXXXXXXX

Recursos y restricciones

- Memoria para *arrays*: 300 bytes.
- Memoria dinámica: nodos de no más de 12 bytes.
- Accesos a disco: 1 acceso secuencial a cada registro de cada archivo más 1 acceso directo al archivo **POSTULANTES.dat** y 2 accesos directos al archivo de **RUBROS.dat**.

NOTA: Se considerará especialmente la adecuada aplicación de la metodología ***top-down***.

02 – 03 - 2013	Cantidad de Hojas:___	Nota:_____ Evaluó Prof:_____	
Apellido y Nombre:_____		Legajo:_____	Curso con:_____

Funciones de biblioteca que puede utilizar, en caso de ser necesario, sin desarrollar. Tenga en cuenta que debe ser preciso con el nombre del módulo y con los parámetros. Los tipos de datos que se definen son genéricos, para su utilización escriba los prototipos con los datos particulares del algoritmo a resolver.

Archivos

Procedure **lecturaEspecial**(var Archivo: tipoArchivo; var Registro: tipoRegistro; var Fin: Boolean).
`//Retorna el registro leído y Fin False, si pudo leer o Fin = True en caso contrario.`
Function **leerEspecial**(var Archivo: tipoArchivo; var Registro: tipoRegistro: boolean.
`//Retorna True si pudo leer o False en caso contrario.`
Function **busquedaBinariaA**(var Archivo: tipoArchivo; N:Entero; clave:tipoInfo): Entero.
`//Retorna en N la referencia al lugar donde se encuentra la clave en el archivo o el valor -1 en caso de no existir.`
Procedure **busqBinA**(var Archivo: tipoArchivo; clave:tipoInfo; var Registro: tipoRegistro).
`Retorna el registro que tiene la clave buscada, la que se supone existe.`

Array

Procedure **ordenarVector**_____(var Vector: tipoVector; N: Entero).
`//Retorna el vector, de tamaño lógico N, ordenado por el campo clave con el que completa el nombre del modulo.`
Procedure **cargarSinRepetir**(var Vector:tipoVector; var Pos, N; var Inserto:Boolean; Clave: tipoInfo).
`//Carga una clave sin repetición en un vector, retorna en Pos, el índice donde lo encontró o lo inserto y en Inserto True, en caso de haberlo insertado.`
Function **busquedaBinariaV**(var Vector: tipoVector; N:Entero; clave:tipoInfo): Entero.
Procedure **busqBinV**(var Vector: tipoVector; N:Entero; clave:tipoInfo; var Pos: Entero).
`//Similar a la BB en archivo modificando la estructura de dato y con el agregado de N que representa el tamaño lógico del vector.`

Estructuras Enlazadas

Procedure **meter**____(var pila: tipoPuntero; valor : tipoInfo).`// inserta un nodo en una pila`
Procedure **sacar**____(var pila: tipoPuntero; var valor : tipoInfo)
`//saca el primer nodo de una pila o una lista`
Procedure **agregar**___(var colaFte, colaFin: tipoPuntero; valor : tipoInfo)`//inserta en una cola`
Procedure **suprimir**(var colaFte, colaFin: tipoPuntero; var valor : tipoInfo)`//saca de una cola`
Procedure **insertaNodo**_____(var lista: tipoPuntero; valor : tipoInfo)`//inserta en una lista`
Procedure **suprimeNodo**(var lista: tipoPuntero; valor : tipoInfo)
`//Busca un nodo con las características de Valor, si lo encuentra lo elimina`
Procedure **buscaInserta**_____(var lista, ptr : tipoPuntero; valor : tipoInfo)
`//busca un nodo con los datos de valor, si no lo encuentra, lo inserta y retorna en ptr la dirección de memoria creada. Si estaba retorna en ptr esa dirección.`
Procedure **insertaPrimero**_____(var lista: tipoPuntero; valor : tipoInfo)
Procedure **insertaDelante**_____(var lista: tipoPuntero; valor : tipoInfo)
Procedure **insertaEnMedio**_____(var lista: tipoPuntero; valor : tipoInfo)
Procedure **insertaAlFinal**_____(var lista: tipoPuntero; valor : tipoInfo).
`//los procedimientos precedentes insertan en las posiciones particulares que sus nombres indican.`
Function **buscaNodo**_____(lista: tipoPuntero; valor : tipoInfo): tipoPuntero
`//busca un nodo con los datos de valor y retorna esa dirección, si no lo encuentra retorna el valor Nulo`