Tema 1: Se tiene una matriz de caracteres letra minúscula. a) Para las primeras 3 filas que tienen una cantidad impar de vocales, se pide obtener el número de la fila que tiene la menor cantidad de consonantes. b) Agregar en un arreglo de caracteres letra minúscula ordenado ascendente (manteniendo su orden en todo momento), las consonantes mayores al caracter letra 'f' de la fila correspondiente al número de fila obtenido en a).
Observaciones: _el tamaño del arreglo es igual a la cantidad de filas de la matriz, _la matriz y el arreglo están precargados, _realizar el programa completo bien modularizado sin métodos de carga e impresión de matriz, ni utilizar estructuras auxiliares.

a) /*pseudocódigo, a la resolución le falta el programa, main, declaraciones y llamadas desde main*/

```
public static int obtener_fila_consonantes(char[][] matriz){
	int cant_filas = CTE3;//las primeras 3 filas que cumplen una condición
	int menor_cantidad = MAXCOLUMNA+1;//inicializa con el peor caso o inexistente
	int fila = 0;
	int menor_fila = -1;
	int cant_vocales, cant_consonantes;
	while(fila<MAXFILA)&&(cant_filas>0){
		cant_vocales = cantidad_vocales(matriz[fila]);
		if (cant_vocales%2==1){
			cant_consonantes = MAXCOLUMNA - cant_vocales;
			if (cant_consonantes<menor_cantidad){
				menor_cantidad = cant_consonantes;
				menor_fila = fila;
			}
			cant_filas- -;
		}
		fila++;
	}
	return menor_fila;
}

public static int cantidad_vocales(char[] arreglo){
	int cant = 0;
	for (int col = 0; col<MAXCOLUMNA; col++){
		if es_vocal(arreglo[col]){
			cant++;
		}
	}
	return cant;
}

public static boolean es_vocal(char valor){
	return ((valor=='a')||(valor=='e')||(valor=='i')||(valor=='o')||(valor=='u'));
}

b)
public static void agregar_consonantes_de_arreglo_de_letras_en_arreglo(char[] arr1; char[] arr2){
	for (int col = 0; col<MAXCOLUMNA; col++){
		if (!es_vocal(arr1[col])&&(arr1[col])>CTEf){//si es consonante mayor a 'f'
			insertar_ordenado(arr2,arr1[col]);
		}
	}
}

public static void insertar_ordenado(char[] arreglo; char valor){
	/*obtener la posición de valor, si es menor al tamaño del arreglo realizar corrimiento a derecha copiar valor en la posición obtenida*/
}
```

Tema 2: Se tiene una matriz que contiene caracteres dígito y caracteres letra minúscula. a) Considerando solo las columnas que tienen caracteres letra minúscula, se pide obtener el número de la columna que tiene la menor cantidad de consonantes menores al caracter 'm'. b) Determinar si en un arreglo dado de caracteres letra minúscula ordenado ascendente, están todas las consonantes incluídas en la columna correspondiente al número de columna obtenido en a).
Observaciones: _el tamaño del arreglo es igual a la cantidad de columnas de la matriz, _la matriz y el arreglo están precargados, _realizar el programa completo bien modularizado sin métodos de carga e impresión de matriz, ni utilizar estructuras auxiliares.

a) /*pseudocódigo, a la resolución le falta el programa, main, declaraciones y llamadas desde main*/

```
public static int obtener_columna_consonantes(char[][] matriz){
	int menor_cantidad = MAXFILA;
	int menor_columna = -1;
	int cant_letras;
	for (int col = 0; col<MAXCOLUMNA, col++){
		cant_letras = cantidad_caracteres_letras(matriz, col);
		if (cant_letras==MAXFILA){ //si solo tiene letras ambas cantidades son iguales
			cant_consonantes = cantidad_consonantes_en_columna_de_caracteres_letras(matriz, col);
			if (cant_consonantes<menor_cantidad){
				menor_cantidad = cant_consonantes;
				menor_columna = col;
			}
		}
	}
	return menor_columna;
}

public static int cantidad_caracteres_letras(char[][] matriz, int col){
	int cant = 0;
	for (int fil = 0; fil<MAXFILA; fil++){
		if ((matriz[fil][col]>='a')&&(matriz[fil][col]<='z')){
			cant++; //cuenta las letras
		}
	}
	return cant;
}

public static boolean es_vocal(char valor){
	return ((valor=='a')||(valor=='e')||(valor=='i')||(valor=='o')||(valor=='u'));
}

public static int cantidad_consonantes_en_columna_de_caracteres_letras(char[][] matriz, int col){
	int cant = 0;
	for (int fil = 0; fil<MAXFILA; fil++){
		if !es_vocal(matriz[fila][col])&&(matriz[fila][col]<CTEm){ //CTEm está definida para 'm'
			cant++; //cuenta las consonantes menores a 'm'
		}
	}
	return cant;
}
```

b) **//por cada consonante de la columna de a) ver si existe en el arreglo**

```
public static boolean existen_consonantes_de_columna_en_arreglo(char[][] matriz; int col, char[] arreglo){
	int fila = 0
	boolean seguir = true;
	while ((fila<MAXFILA)&&(!seguir)){ //no era necesario agregar seguir en el while
		if ((!es_vocal(matriz[fila][col]))&&(obtener_pos(arreglo,matriz[fil][col])==-1)){
			seguir = false; //si es consonante y no está corta la búsqueda
		}
		else{
```

```
                        fila++;
                    }
                }
                return seguir; //retorna true si cada vez que había una consonante la encontró en arreglo
}
/*obtener la posición de un valor en un arreglo ordenado: en este caso retorna -1 si no está*/
```