Para encontrar la primera presencia de una palabra en un texto se dispone de tres algoritmos: el método de búsqueda directa de cadena, el método de Knuth-Morris-Pratt (KMP) y el algoritmo de Booyer-Moore (BM).

a) Indicar cuál de ellos utilizaría para encontrar una palabra en un texto con 10000 caracteres. ¿Utilizaría el mismo si el texto fuera de 50 caracteres?. Explique su elección en cada caso razonando comparativamente, es decir, mostrando sus ventajas respecto a los otros métodos.

b) Explicar claramente como se realiza la comparación de caracteres en el algoritmo de Boyer-Moore y como se calcula su tabla de distancias. Implementar en modula-2 el segmento de código para calcular dicha tabla.

El número de operaciones necesarias con cada uno de los algoritmos viene por el orden de las comparaciones realizadas más la precompilación del patrón (si la hubiera), lo cual se puede expresar como:

	Operaciones de comparación	Operaciones de Precompilación
Búsqueda directa	M * N	0
Knuth-Morris-Pratt (KMP)	M + N	M
Boyer-Moore (BM)	N/M (óptimo) ≤Nº ≤ N(pésimo)	128 (tabla de caracteres)

donde N es el tamaño del texto y M la longitud de la cadena a buscar. Considerando ambos casos, esto es, de texto con 10.000 o con 50 caracteres, se tiene que

	10 000 caracteres			50 caracteres		
	Comparac.	Precomp.	Total	Comparac.	Precomp.	Total
B.Dir.	10000M	0	10000M	50M	0	50M
KMP	10000+M	M	10000+2M	50+M	M	50+2M
BM	(10000/M,10000)	128	$(10^4/M,10^4)$+128	(50/M,50)	128	(50/M,50)+128

Si la longitud del texto fuera de 10.000 caracteres, sería mejor el algoritmo de BM ya que es el que menor coste presenta en comparaciones. Sin embargo, cuando la cadena consta tan sólo de 50

caracteres, el coste de la precompilación de la tabla de caracteres tiene mayor importancia que el número de comparaciones, y por tanto su rendimiento es superado por el algoritmo KMP.
La búsqueda directa no se considera en ningún caso, por su elevado coste, pese a no utilizar ningún tipo de precompilación.

Las comparaciones se comienzan a realizar por la derecha del patrón, esto es, se comienza comparando el último carácter del patrón con la cadena de texto. Si hay concordancia, se compara el penúltimo carácter del patrón con el anterior de la cadena. Y así hasta alcanzar el inicio del patrón. Cuando haya una inconcordancia se mira si ese carácter está en el patrón: si está, se corre el patrón a la derecha hasta que coincidan ambos caracteres; si no está, el índice de búsqueda en el texto se desplaza la longitud total del patrón más uno, es decir, se corre el patrón entero uno más de la última inconcordancia.
Ejemplo:

```
Este calor sofocante daba mayor valor a quienes salían a la calle
valor
     valor
          valor
           valor
                valor
                     valor
                          valor
                               valor
                                valor
```

En el primer caso, como el patrón no coincide, se desplaza todo el patrón. Esto conduce a la coincidencia del último carácter, y comparando hacia atrás, coinciden hasta el segundo. Se vuelve a desplazar el patrón , coincidiendo con la “o” del texto. En este caso se desplaza el patrón hasta que la última “o” del patrón coincida con la del texto. Al no haber concordancia, se desplaza de nuevo.
El código en Modula-2 que realiza la precompilación del patrón se puede escribir como

```
FOR ch:=0C TO 177C DO d[ch]:=M END;
FOR j:=0 TO M-2 DO d[p[j]]:=M-j-1 END;
```

El primer bucle llena la tabla de desplazamientos de cada carácter con la longitud de la cadena. Posteriormente, se recorren todos los caracteres del patrón (segundo bucle) y se calcula su distancia al final de la cadena. Si hay caracteres repetidos en el patrón, la tabla almacenará la menor de las distancias desde el carácter repetido hasta el final de la cadena.