NOMBRE Y APELLIDOS: GREGORIO CORPAS PRIETO
GRUPO: GP8

1. ENUNCIADO:
   ▶ Diseña la función evaluaPolinomio() que calcule el resultado de un polinomio guardado en una lista enlazada, dependiendo de un valor de x.

   ▶ El programa principal provee de dicha lista con los monomios y pide al usuario un valor flotante para x, a partir del cual desarrollar el polinomio y retornar el valor resultante.

2. DATOS DE ENTRADA:
   ▶ El programa pedirá al usuario que introduzca por teclado el valor de x.
   - Nombre: x
   - Correspondencia: Valor de la incógnita x en el polinomio
   - Tipo de dato: Flotante

3. RESULTADOS O SALIDA:
   ▶ La función retornará la sumatoria de todos los elementos del polinomio.
   - Nombre: Retorno
   - Correspondencia: Suma de todos los monomios del polinomio
   - Tipo de dato: Flotante

   ▶ Una vez obtenido el valor, este será retornado al programa principal.

4. DATOS AUXILIARES:
   ▶ Haremos uso de una librería de cálculo matemático que incluye un método para realizar potencias.

5. DESCRIPCIÓN DEL ALGORITMO:
   ▶ Inicializaremos la variable retorno a 0 el fin de comenzar la función.

   ▶ Guardaremos la cabeza de la lista polinomio en una variable auxiliar.

   ▶ Recorreremos iterativamente la lista mientras que el siguiente elemento de auxiliar sea distinto de NULL (mientras que no lleguemos al final de la lista). Para esto modificaremos el valor de auxiliar al del siguiente elemento en cada iteración.

   ▶ En cada iteración se calcula y añade a retorno el resultado del monomio actual, multiplicando el coeficiente almacenado en auxiliar, por la potencia de el valor X (parámetro de esta función) elevado al exponente almacenado en auxiliar.

   'retorno+=auxiliar->coeficiente * potencia(x, auxiliar->exponente)'

   ▶ Devolvemos la variable retorno y termina la función.

## 6. PSEUDOCÓDIGO

```
PROGRAMA PRINCIPAL:

        FLOTANTE X, RESULTADO
        MONOMIO * POLINOMIO

        IMPRIME ( INTRODUCE EL VALOR DE X CON EL QUE EVALUAR EL POLINOMIO: )
        LEE ( X )

        RESULTADO = EVALUAPOLINOMIO ( POLINOMIO, X )

        IMPRIME ( EL RESULTADO DEL POLINOMIO USANDO X = [X] ES [RESULTADO] )

FIN PROGRAMA PRINCIPAL

EVALUAPOLINOMIO ( MONOMIO * POLINOMIO, ENTERO X ):

        MONOMIO * AUXILIAR = POLINOMIO
        FLOTANTE RETORNO

        MIENTRAS ( AUXILIAR != NULL )
                RETORNO += AUXILIAR->COEFICIENTE * POTENCIA(X, AUXILIAR->EXPONENTE)
                AUXILIAR = AUXILIAR->SIGUIENTE
        FIN MIENTRAS

        RETURN RETORNO

FIN  EVALUAPOLINOMIO
```

1. ENUNCIADO:
   ▶ Diseña la función buscaContenedor() del ejercicio 6

2. DATOS DE ENTRADA:
   ▶ Cima de la pila donde buscar el dato
      - Nombre: cima
      - Correspondencia: Ultimo registro apilado o cabeza de la lista
      - Tipo de dato: struct nodo **
      - Restricciones: Ninguna

   ▶ Valor a buscar en la pila
      - Nombre: valor
      - Correspondencia: valor a buscar en la pila
      - Tipo de dato: int
      - Restricciones: Ninguna

3. RESULTADOS O SALIDA:
   ▶ El retorno será 1 si se encontró el valor o 0 si no se encontró

4. DATOS AUXILIARES:
   ▶ Necesitaremos una variable de tipo struct nodo * para crear una pila auxiliar en la que iremos guardando los nodos desapilados de la pila principal.

   ▶ También necesitaremos la variable bandera que nos dirá si se encontró el elemento en la pila o no.

5. DESCRIPCIÓN DEL ALGORITMO:
   ▶ Se seteará la bandera de encontrado a 0, y esta no se modificará hasta encontrar el elemento buscado.

   ▶ Haciendo uso de la librería de funciones facilitada en moodle, recorreremos la pila mientras que se cumplan 2 premisas:
      - El elemento no haya sido encontrado
      - La función pilaVacia(*cima) retorne que aún no está vacía

   ▶ Dentro de este recorrido iremos desapilando de la pila los elementos y apilandolos en la pila auxiliar haciendo uso de las funciones push y pop.

   ▶ Acabada la búsqueda, encontrado o no el elemento, se debe retornar todo lo desapilado a la pila principal. Por tanto se recorrerá la pila auxiliar mientras que la función pilaVacía(cima_aux) no indique lo contrario.

   ▶ Reapilados los elementos y con el resultado de la búsqueda, solo nos queda devolver el valor mediante un return.

## 6. PSEUDOCÓDIGO

```
PROGRAMA PRINCIPAL:

        STRUCT NODO * PILA
        ENTERO BUSQUEDA

        IMPRIME( INTRODUCE EL VALOR A BUSCAR EN LA PILA )

        LEE ( BUSQUEDA )

        SI ( BUSCACONTENEDOR ( &PILA , BUSQUEDA ) == 1 )

                IMPRIME( ELEMENTO [BUSQUEDA] ENCONTRADO EN LA PILA )

        SINO

                IMPRIME( ELEMENTO [BUSQUEDA] NO ENCONTRADO EN LA PILA)

        FIN SI

FIN PROGRAMA PRINCIPAL

BUSCACONTENEDOR ( STRUCT NODO ** CIMA, ENTERO BUSQUEDA):

        STRUCT NODO * CIMA_AUX
        INT ENCONTRADO = 0

        //Mientras no encontramos el elemento y no llegamos al final de la pila
        MIENTRAS ( ( PILAVACIA( *CIMA ) == 0 ) && ( ENCONTRADO == 0 ) )

                SI ( (*CIMA)->NUMERO == BUSQUEDA)
                        ENCONTRADO = 1
                FIN SI

                PUSH ( &CIMA_AUX, POP ( CIMA ) ) //Desapilaremos en la pila auxiliar

        FIN MIENTRAS

        //Mientras queden elementos en la pila auxilar
        MIENTRAS ( PILAVACIA( CIMA_AUX ) != 1 )

                PUSH ( CIMA , POP ( &CIMA_AUX ) ) //Desapilaremos en la pila principal

        FIN MIENTRAS

        RETURN ENCONTRADO

FIN BUSCACONTENEDOR
```