**Universidad de las fuerzas armadas ESPE**

**Grupo1**

**Problema 2.1.3 –Vector con término general dado**

Requisitos funcionales:

1. El programa debe leer un número entero n (cantidad de componentes del vector), asegurando que n ≤ 100.
2. Debe calcular los n términos de la sucesión vk=k2+3v\_k = k^2 + 3vk​=k2+3 (para k=1k = 1k=1 a nnn).
3. Los resultados deben ser almacenados en un vector vec[], de forma que vec[i] = (i + 1)^2 + 3.
4. El programa debe mostrar en pantalla cada componente del vector mientras la calcula.

**Problema 2.1.4 – Comprobar si dos valores pertenecen a un vector**

Requisitos funcionales:

1. El programa debe leer dos números enteros desde teclado.
2. Debe contar con un vector de enteros predefinido de tamaño 15 (puede ser ingresado o definido directamente).
3. Se debe recorrer el vector y comparar cada elemento con los dos números ingresados.
4. El programa debe usar dos variables banderas (inicializadas en 0), una para cada número.
5. Si algún número coincide con un valor del vector, su respectiva bandera se pone en 1.
6. Al final del recorrido, si ambas banderas son 1, se debe indicar que ambos números están en el vector.

**Problema 2.1.5 – Vector de factoriales**

Requisitos funcionales:

1. El programa debe contar con un vector Vec que contenga los primeros 15 números naturales (del 1 al 15).
2. Debe calcular la factorial de cada número contenido en Vec.
3. Los resultados deben almacenarse en un nuevo vector fact[].
4. El programa debe mostrar en pantalla el vector fact[] con los factoriales calculados.
5. Se debe utilizar un bucle externo para recorrer Vec, y un bucle interno para calcular cada factorial.
6. Advertencia técnica: Si se usa un tipo de dato entero normal (int), el cálculo de la factorial de 13 en adelante puede fallar. Se recomienda usar un tipo como unsigned long int para evitar errores por desbordamiento.

**Problema 2.1.6 – Ordenación de un vector**

Requisitos funcionales:

1. El programa debe ordenar un vector de 10 componentes en orden de mayor a menor.
2. Se debe asumir que el vector ya está leído y almacenado en memoria.
3. El método de ordenación debe ser iterativo, usando dos bucles anidados.
4. El algoritmo debe comparar cada elemento con los siguientes y realizar intercambios si están fuera de orden.
5. El proceso se repite hasta que el vector esté completamente ordenado (tipo bubble sort descendente).
6. Se deben comparar pares de elementos y realizar los intercambios necesarios para colocar el mayor valor al inicio del vector.