**Universidad Nacional de La Matanza**

Depto. de Ingeniería e Investigaciones Tecnológicas



**PROGRAMACIÓN**

**AVANZADA**

TRABAJO PRÁCTICO 1

Integrantes:

|  |  |  |
| --- | --- | --- |
| APELLIDO, Nombre | E-mail | DNI |
| AMORUSO, Sergio Federico | sergioamoruso91@gmail.com | 36593815 |
| BRUDE, Alejandro Gabriel | [alejandrobrude@gmail.com](mailto:alejandrobrude@gmail.com) | 33908097 |
| CABRAL, Rodrigo Ariel | [cabralrodrigoariel@gmail.com](mailto:cabralrodrigoariel@gmail.com) | 35103809 |
|  |  |  |

PRIMER CUATRIMESTRE – AÑO 2017

**RESULTADO DEL POLINOMIO: LOTE DE PRUEBAS**

A continuación, se muestran los casos de prueba realizados sobre las distintas versiones del programa que evalúa polinomios por medio de múltiples métodos. En cada caso se presenta en primer lugar la descripción, en segundo lugar, la entrada, seguido de la salida que se espera en base a dicha entrada, y por último qué fue lo que se obtuvo al ejecutar cada versión.

**CASO 1: PolinomioTrivial**

Se busca calcular el resultado de x = 2.

|  |
| --- |
| **ENTRADA.IN** |
| 5 |
| 2 |
| 9 |
| 8 |
| 0 |
| 5 |

|  |
| --- |
| **SALIDA.OUT** |
| 219 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **Evaluar**  **Pow** | **Evaluar**  **MSucesivas** | **Evaluar**  **Recursiva** | **Evaluar**  **RecursivaPar** | **Evaluar**  **Horner** | **Evaluar**  **ProgDinamica** | **Evaluar**  **Mejorada** |
| **OUT** | 219 | 219 | 219 |  |  |  |  |

**CASO 2: RaizTriple**

Se busca calcular el resultado de x = 2 raíz triple.

|  |
| --- |
| **ENTRADA.IN** |
| 4 |
| 2 |
| 1 |
| -6 |
| 12 |
| -8 |

|  |
| --- |
| **SALIDA.OUT** |
| 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **Evaluar**  **Pow** | **Evaluar**  **MSucesivas** | **Evaluar**  **Recursiva** | **Evaluar**  **RecursivaPar** | **Evaluar**  **Horner** | **Evaluar**  **ProgDinamica** | **Evaluar**  **Mejorada** |
| **OUT** | 0 | 0 | 0 |  |  |  |  |

**CASO 3: RaizDoble**

Se busca calcular el resultado de x = 2 raíz doble.

|  |
| --- |
| **ENTRADA.IN** |
| 3 |
| 2 |
| 1 |
| -4 |
| 4 |

|  |
| --- |
| **SALIDA.OUT** |
| 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **Evaluar**  **Pow** | **Evaluar**  **MSucesivas** | **Evaluar**  **Recursiva** | **Evaluar**  **RecursivaPar** | **Evaluar**  **Horner** | **Evaluar**  **ProgDinamica** | **Evaluar**  **Mejorada** |
| **OUT** | 0 | 0 | 0 |  |  |  |  |

**CASO 4: Fatiga01**

Se busca determinar si se produce un fallo con valores grandes y x = 100.

|  |
| --- |
| **ENTRADA.IN** |
| (x-5)^10 |
|  |
|  |
|  |

|  |
| --- |
| **SALIDA.OUT** |
|  |
|  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **Evaluar**  **Pow** | **Evaluar**  **MSucesivas** | **Evaluar**  **Recursiva** | **Evaluar**  **RecursivaPar** | **Evaluar**  **Horner** | **Evaluar**  **ProgDinamica** | **Evaluar**  **Mejorada** |
| **OUT** |  |  |  |  |  |  |  |

**CASO 5: Raiz0 //verificar, no está agregado el caso de que el P[x] no tenga termino independiente.**

Se busca calcular el resultado de x = 0 raíz.

|  |
| --- |
| **ENTRADA.IN** |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| **SALIDA.OUT** |
|  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **Evaluar**  **Pow** | **Evaluar**  **MSucesivas** | **Evaluar**  **Recursiva** | **Evaluar**  **RecursivaPar** | **Evaluar**  **Horner** | **Evaluar**  **ProgDinamica** | **Evaluar**  **Mejorada** |
| **OUT** |  |  |  |  |  |  |  |

**CASO 6: RaizIrracional**

Se busca calcular el resultado de x = (2)^(1/2) raíz.

|  |
| --- |
| **ENTRADA.IN** |
| 4 |
| (2)^(1/2) |
| 1 |
| -3 |
| 1 |
| 7 |

|  |
| --- |
| **SALIDA.OUT** |
|  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 1 | SI 1 | SI 1 | NO | SI | SI 1 | SI 1 | SI 1 | SI 1 |
| 1 | 1 | 1 |  | 1 1 | 1 | 1 | 1 | 1 |

**CASO 7: DetectaTodasCombinaciones3Digitos**

Se busca que se detecten todas las combinaciones posibles de un número de tres dígitos distintos.

|  |
| --- |
| **ENTRADA.IN** |
| 23 |
| 12301320231021303210312 |
| 3 |
| 123 |

|  |
| --- |
| **SALIDA.OUT** |
| SI 6 |
| 1 5 9 13 17 21 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 6 | SI 3 | SI 6 | SI 5 | SI | SI 7 | SI 2 | SI 6 | SI 6 |
| 1 5 9 13 17 21 | 1 9 21 | 1 5 9 13 17 21 | 1 5 9 13 17 | 6 1 5 9 13 17 21 | 1 5 9 13 15 17 21 | 1 13 | 1 5 9 13 17 21 | 1 5 13 9 21 17 |

**CASO 8: DetectaTodasCombinaciones4Digitos**

Se busca que se detecten todas las combinaciones posibles de un número de cuatro dígitos distintos.

|  |
| --- |
| **ENTRADA.IN** |
| 119 |
| 12340124301324013420142301432021340214302314023410241302431031240314203214032410341203421041230413204213042310431204321 |
| 4 |
| 1234 |

|  |
| --- |
| **SALIDA.OUT** |
| SI 24 |
| 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 24 | SI 4 | SI 8 | SI 23 | SI | SI 24 | SI 6 | SI 24 | SI 24 |
| 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 | 1 46 81 91 | 1 26 36 46 71 81 91 116 | 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 | 24 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 | 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 | 1 21 41 61 81 101 | 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 | 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 |

**CASO 9: DetectaTodasCombinaciones4DigitosConRepetido**

Se busca que se detecten todas las combinaciones posibles de un número de cuatro dígitos que contiene dos dígitos iguales.

|  |
| --- |
| **ENTRADA.IN** |
| 59 |
| 22130223102123021320232102312012230123201322032210321203122 |
| 4 |
| 2213 |

|  |
| --- |
| **SALIDA.OUT** |
| SI 12 |
| 1 6 11 16 21 26 31 36 41 46 51 56 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 12 | SI 4 | SI 8 | SI 11 | SI | SI 12 | SI 3 | SI 12 | SI 12 |
| 1 6 11 16 21 26 31 36 41 46 51 56 | 1 16 41 46 | 1 6 16 26 31 41 46 56 | 1 6 11 16 21 26 31 36 41 46 51 | 12 1 6 11 16 21 26 31 36 41 46 51 56 | 1 6 11 16 21 26 31 36 41 46 51 56 | 1 21 41 | 1 6 11 16 21 26 31 36 41 46 51 56 | 1 6 11 16 21 26 31 36 41 46 51 56 |

**CASO 10: QueNoDetectePorSumaDeDigitos**

Se busca que no detecte comparando por cantidad y suma de dígitos a buscar.

|  |
| --- |
| **ENTRADA.IN** |
| 2 |
| 04 |
| 2 |
| 31 |

|  |
| --- |
| **SALIDA.OUT** |
| NO |
|  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | NO | NO | NO | NO | NO | SI 1 | NO | NO | SI 12 |
|  |  |  |  |  | 1 |  |  |  |

**CASO 11: DetectaCombinacionesSecuenciaRepetida**

El programa debe tomar todas las combinaciones del segundo número cuando aparece repetido en el mismo orden dentro del primero.

|  |
| --- |
| **ENTRADA.IN** |
| 20 |
| 12341234123412341234 |
| 4 |
| 1234 |

|  |
| --- |
| **SALIDA.OUT** |
| SI 17 |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 17 | SI 17 | SI 17 | SI 16 | SI | SI 17 | SI 5 | SI 17 | SI 17 |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | 17 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 1 5 9 13 17 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | 1 5 9 13 17 2 6 10 14 3 7 11 15 4 8 12 16 |

**CASO 12: DetectaValoresContiguosCompartidos**

|  |
| --- |
| **ENTRADA.IN** |
| 14 |
| 12343213423124 |
| 4 |
| 1234 |

|  |
| --- |
| **SALIDA.OUT** |
| SI 6 |
| 1 4 6 7 9 11 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 6 | SI 1 | SI 2 | SI 5 | SI | SI 6 | SI 2 | SI 6 | SI 6 |
| 1 4 6 7 9 11 | 1 | 1 4 | 1 4 6 7 9 | 6 1 4 6 7 9 11 | 1 4 6 7 9 11 | 1 9 | 1 4 6 7 9 11 | 1 7 6 11 9 4 |

**CASO 13: DetectaCombinacionesExtensas**

|  |
| --- |
| **ENTRADA.IN** |
| 50 |
| 01245897631125430698715973468098765412309630258741 |
| 10 |
| 0123456789 |

|  |
| --- |
| **SALIDA.OUT** |
| SI 7 |
| 1 12 13 30 31 32 41 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EXE** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| **OUT** | SI 7 | NO | NO | SI 6 | SI | SI 7 | SI 3 | SI 7 |  |
| 1 12 13 30 31 32 41 |  |  | 1 12 13 30 31 32 | 7 1 12 13 30 31 32 41 | 1 12 13 30 31 32 41 | 1 31 41 | 1 12 13 30 31 32 41 | Se queda colgado |

**NÚMERO ESCONDIDO: RESULTADO DE LAS PRUEBAS**

|  |  |
| --- | --- |
| **PROGRAMA** | **RESULTADO** |
| **A** | No se encontraron bugs. |
| **B** | Encuentra válidos sólo los números combinándolos de forma circular hacia la derecha pero no los alterna.  Ejemplo:  En el caso del número 123, sólo detecta las combinaciones 123, 231 y 312. |
| **C** | Encuentra sólo 2 combinaciones por cada dígito del número dado. Una de ellas la genera rotando de forma circular una posición hacia izquierda. La otra, partiendo de la anterior, fija el primer dígito y luego da vuelta los restantes.  Ejemplo:  En el caso del número 1234 podrían ser 2341 y fijando el número 2 la otra sería 2143. |
| **D** | Recorre vector con primer número hasta la posición (cantidadDigitosPrimerNumero - 2), en vez de recorrer hasta la posición (cantidadDigitosPrimerNumero - 1). |
| **E** | Muestra la salida con el formato incorrecto: la cantidad de veces que el primer número contiene al segundo se encuentra en la segunda línea en vez de encontrarse en la primera. |
| **F** | Detecta una combinación correcta cuando la cantidad de dígitos del número que busca coincide y la suma de todos ellos también, sin considerar si son los mismos dígitos. |
| **G** | No toma en cuenta las combinaciones entrelazadas, si la cuenta falla, ignora el valor y pasa al siguiente. |
| **H** | No se encontraron bugs. |
| **I** | Evidentemente el programa forma de manera interna, todas las combinaciones posibles, y las compara con la cadena de comparación. Al poner números extensos el programa se tilda y no devuelve salida. |