Parcial 2 - Algoritmos I Taller: Tema B

Debés entregar el código en el formulario en el que completaste tus datos personales. El texto de cada ejercicio por separado con todo el código necesario para que ese ejercicio compile y pueda probarse. Además debés subir el archivo con el código fuente de cada ejercicio.

Para compilar un archivo .c escribir en la terminal en la carpeta donde esta el archivo:
`$> gcc -Wall -Wextra -std=c99 miarchivo.c -o miprograma`
Para ejecutar escribir:
`$> ./miprograma`

Ejercicio 1

Considerar el tipo `silksong_enemy_t`

```
C/C++
typedef struct _senemy_t {
    int dano;
    int vida;
    char condicion; // 'A' o 'N', dependiento si es agresivo o neutral
    int recompensa;
} silksong_enemy_t;
```

que es una estructura que guarda la información acerca del daño de un enemigo en Pharloom, su cantidad de vida, si es agresivo o neutral y la cantidad de recompensa que da al eliminarlo. Se debe definir la función

```
C/C++
int resistencia(silksong_enemy_t enemy, int vida_propia)
```

que calcula la cantidad de golpes que podemos resistir del enemigo `enemy` teniendo en cuenta nuestra cantidad de vida `vida_propia`.

Luego en la función `main()` definir una variable `skull_tyrant` de tipo `silksong_enemy_t` que contenga la información de ese enemigo: daño 2, vida 400, es agresivo y la recompensa es de 70 rosarios. Además se debe solicitar al usuario ingresar un entero y guardarlo en una variable `vp`. Por último, en la función `main()` usar la función `resistencia()` para determinar la cantidad de golpes que podemos resistir del enemigo si nuestra vida es `vp`. Por ejemplo, si el usuario ingresa `5` debe mostrar el mensaje `La cantidad máxima golpes que podemos resistir del enemigo Skull Tyrant es 3`. **Incluir al final del archivo al menos un ejemplo de ejecución como comentarios.**

Ejercicio 2

Definir la función `int sum_intercalado(int n)` para ello considerar el siguiente algoritmo:

```
Textproto
Const N : Int;
Var f, n : Int;
{P : N >= 0}
f, n := 0, 0
do n ≤ N →
  n, f := n + 2, f + n
od
{Q : f = ⟨Σi : 0 ≤ i ≤ N ∧ i mod 2 = 0: i⟩}
```

Verificar la pre-condición usando `assert()`. Luego en la función `main()` pedir al usuario un entero y luego mostrar el resultado de la función por pantalla. **Incluir al final del archivo al menos un ejemplo de ejecución como comentarios.**

Ejercicio 3

Definir la función `bool alguno_par_pos_pares(int tam, int arr[])` implementando el siguiente algoritmo:

```
Textproto
Const N : Int, A : array [0, N) of Int;
Var n: Int;
Var p: Bool;
{P : N >= 0}
n, p := 0, false
do n < N →
  n, p := n + 2, p ∨ (A.n mod 2 = 0)
od
{Q : p = ⟨ Ei : 0 ≤ i < N ∧ i mod 2 = 0 : A.i mod 2 = 0⟩}
```

Verificar la pre-condición usando `assert()`. Luego completar la siguiente función `main()`:

```
C/C++
#define N 11

int main(void) {
    int valores[N] = {1, 0, 3, -2, 5, 12, 7, 2, 1, 15, 0};
    /* COMPLETAR */
    return 0;
}
```

y mostrar un mensaje que diga `Al menos un elemento en las posiciones pares del arreglo es par` en caso que la función `alguno_par_pos_pares()` aplicada al arreglo `valores` devuelva `true`, y en caso contrario debe mostrar el mensaje `No hay elementos pares en las posiciones pares del arreglo`. **Incluir al final del archivo al menos un ejemplo de ejecución como comentarios.**