1. Como es bien sabido, la representación numérica binaria de tipo signo/exponente/mantisa, depende de la creatividad o precisión que desee el programador. Analicemos qué sucede con un sistema de numeración binario a $n > 0$ (entero) bits, para esto elige un valor par entre 10 y 20, crea el sistema de numeración entero similar al estudiado en clase, (la mantisa debe ser, al menos, la mitad del número elegido). Describe tu sistema de numeración, ¿Cuál es el menor y mayor de los números representados? Presenta un número que no se pueda representar en tu sistema seleccionado. Describe tu proceso. (4 pts) NOTA: punto extra si además se describe un sistema fraccionario que tenga sentido.

Demostración. Usaremos 16 bits en total. 1 para el signo, 3 para el exponente, 12 para la mantisa. Esto nos genera números en el rango

$$[-1 * 2^7 * (2^{12} - 1), 2^7 * (2^{12} - 1)] = [-524160, 524160]$$

□

2. Consideremos la siguiente sucesión $\{x_n\}_{n\in\mathbb{N}} \subset \mathbb{R}$ definida de manera recurrente como: (4 pts)

$$x_2 = 2,$$
$$x_{n+1} = 2^{n-\frac{1}{2}}\sqrt{1 - \sqrt{1 - 4^{1-n}x_n^2}}$$

   (a) Escribe un programa para encontrar los primeros n términos de la sucesión.

   (b) Aplica tu programa para encontrar los términos de la sucesión cuando $n = 6, 7, 8, 9, 10, 12, 18, 20$, ¿a qué valor tiende la sucesión? (ésta sucesión tiende a un valor distinto de cero).

   (c) Encuentra los términos 50 y 100, ¿qué sucede con la sucesión? Grafica los términos de la sucesión desde 2 hasta 100. Explica detalladamente el por qué del comportamiento que tiene la computadora al calcular estos términos.

Listing 1: Calcular epsilon de la máquina

Demostración.
```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define SQRT2 1.4142135624

double xn(int n) {
  double x = 2.0;
  double two_pow_n_minus_one = SQRT2;
  double two_pow_two_minus_two_n = 1.0;
  printf("x_2 = %lf\n", x);
  for (int i = 3; i < n + 2; i++) {
    x = (two_pow_n_minus_one *= 2) *
        sqrt(1 - sqrt(1 - (two_pow_two_minus_two_n /= 4) * x * x
            ));
    printf("x_%d = %lf\n", i, x);
  }
```

```
  return x;
}

int main(int argc, char *argv[]) {
  if (argc != 2) {
    fprintf(stderr, "Invalid arguments. Usage: ./p2 (terms)");
    exit(1);
  }
  int terms = atoi(argv[1]);
  xn(terms);
  return 0;
}
```

□

3. Al número representable inmediatamente después de la unidad se le conoce como el epsilón de la computadora machine el cual nos permite calcular el número real representable inmediatamente posterior; ésto se obtiene al multiplicar el epsilón por el real y sumarlo a ese real. Utilizando el siguiente algoritmo, crea un código que calcule el machine de tu computadora, sólo hace falta indicar el tipo de variable que es "epsilon" (float o doble). El último valor impreso corresponde al epsilon de la computadora. (2 pts)

Demostración. Después de 52 iteraciones, obtenemos que

$$\epsilon = 0.00000000000000002220446 = 2.22046 \times 10^{-16} \approx 2^{-52}$$

.

Listing 2: Calcular epsilon de la máquina

```
#include <stdio.h>

int main() {
  double epsilon = 1;
  int iter = -1;
  while (1+epsilon > 1) {
    iter++;
    printf("%.22f\n", epsilon);
    epsilon /= 2;
  }
  printf("%d\n", iter);
}
```

□