4.1 Simule a execução dos seguintes fragmentos de programas e indique os resultados.

(a)
```
int i = 1;
while (i <= 128) {
   printf("%d ", i);
   i *= 2;
}
```

(b)
```
int i = 0;
while (i*i < 25)
  i ++;
printf("%d\n", i);
```

(c)
```
int n = 9384;
while (n > 0) {
  printf("%d\n", n%10);
  n /= 10;
}
```

(d)
```
int n = 9384;
do {
   printf("%d\n", n);
   n /= 10;
} while (n > 0);
```

4.2 Considere o programa para calcular comissões de transferência apresentado na aula teórica 4. Acrescente um ciclo ao programa de forma a que este leia repetidamente valores e imprima as comissões correspondentes; deve terminar quando for introduzido o valor 0.

4.3 Escreva um programa que lê uma sequência de inteiros terminada pelo valor 0 e determina o *máximo* (i.e. maior valor da sequência). *Sugestão*: modifique o exemplo apresentado na aula teórica 6 para somar uma sequência de números.

4.4 Escreva um programa que lê uma sequência de valores em vírgula-flutuante terminada pelo valor 0 e calcula a sua *média aritmética* (i.e. a soma dos valores a dividir pela sua contagem). *Sugestão*: modifique o exemplo apresentado na aula teórica 6 para somar uma sequência de números.

▷ **4.5** Escreva uma definição da função `int soma_divisores(int n)` que calcula a soma dos divisores de n inferiores a ele próprio. Por exemplo: a soma dos divisores de 12 é 16 porque os divisores de 12 menores que 12 são $\{1, 2, 3, 4, 6\}$ e $1 + 2 + 3 + 4 + 6 = 16$. *Sugestão*: modifique o exemplo apresentado na aula teórica 7 para testar se um número é primo.

4.6 No nosso calendário Gregoriano os *anos bissextos* têm 366 dias em vez dos normais 365. As regras para determinar se um ano é bissexto são:

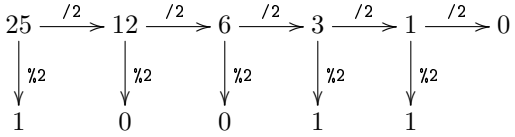
- São bissextos todos os anos múltiplos de 400 (e.g., 1600, 2000, 2400, 2800...);
- São bissextos todos os múltiplos de 4, exceto se forem múltiplos de 100 mas não de 400 (e.g., 1996, 2000, 2004, 2008, 2012, 2016...);
- Não são bissextos todos os demais anos.

Defina uma função `int bissexto(int n)` que verifica estas condições para um ano `n`; o resultado deve ser 1 se o ano for bissexto e 0 caso contrário.

▷ **4.7** Escreva uma função `int prox_bissexto(int n)` que calcula o próximo ano bissexto a partir do ano dado; se `n` for um ano bissexto, então o resultado deve ser `n`. Exemplos: `prox_bissexto(1980)` retorna 1980; `prox_bissexto(2017)` retorna 2020.

Sugestão: utilize um ciclo `while` começando no ano dado e a função do exercício 4.6 para determinar se um ano é bissexto.

4.8 Generalizando o exemplo apresentado na aula teórica 6, escreva um programa que lê um inteiro não-negativo e imprime os seus dígitos binários. Podemos obter a representação de um inteiro positivo n numa base b fazendo divisões sucessivas por b até obter quociente 0; os restos das divisões são os "dígitos" (de 0 a $b-1$).
Exemplo: obtemos a representação em binário de 25 fazendo 4 divisões sucessivas por 2:



Assim, 25 decimal é representado em binário por 11001. Note que os dígitos são obtidos pela ordem do menos significativo para o mais significativo.

4.9 Escreva um programa que lê dois valores inteiros de numerador e denominador e escreve a fração correspondente simplificada. Exemplo:

```
Numerador: 56
Denominador: 32
A fração 56/32 é equivalente a 7/4
```

Sugestão: Para simplificar uma fração p/q basta dividir o numerador e denominador pelo máximo divisor comum de p, q. Utilize o método de Euclides apresentado na aula teórica 7 para calcular o m.d.c.

4.10 Calcular uma raiz quadrada $\sqrt{a}$ para $a > 0$ corresponde a encontrar a solução positiva da equação $x^2 - a = 0$. Podemos calcular aproximadamente a solução usando o "método Babilónico" (é um caso particular do *método de Newton* para resolução de equações):

1. Escolhemos uma primeira aproximação (por exemplo, $x_0 = a/2$);
2. Usando x_0 calculamos uma aproximação melhor $x_1 = \frac{1}{2}(x_0 + a/x_0)$
3. Repetimos este processo calculando $x_2, x_3, \ldots$ etc. até atingir a precisão pretendida pela a recorrência $x_{n+1} = \frac{1}{2}(x_n + a/x_n)$.

Escreva um programa que lê o valor de a e calcula 10 aproximações a $\sqrt{a}$ pelo método acima imprimindo os valores sucessivos. *Sugestão:* utilize um ciclo `for`.

4.11 Escreva um programa que lê um inteiro não-negativo e imprime a lista de fatores primos (i.e. uma versão do comando `factor` do sistema Linux). Exemplo:

```
Numero inteiro: 24
24 : 2 2 2 3
```

Temos então que $24 = 2 \times 2 \times 2 \times 3$.
Sugestão: utilize um ciclo em que faz divisões sucessivas por inteiros crescentes começando por 2, depois 3, etc. Note que se dividir tantas vezes quanto possível por um inteiro já não será possível dividir pelos seus múltiplos, logo vamos apenas obter os fatores primos tal como pretendido.