Zadanie 1

Funkcja sprawdza czy podany numer PESEL jest prawidłowy. Jeżeli nie, zwraca wartość **`0`** i wyświetla komunikat **`'Niepoprawny PESEL'`**, jeśli tak, zwraca wartość **`1`** i wyświetla komunikat **`'Podał/a Pan/i prawidłowy numer PESEL'`** w zależności od płci (parzystości 10. cyfry numeru PESEL).

```
function isValid = validate(pesel)
    pesel = num2str(pesel);

    if ~all(isstrprop(pesel, 'digit'))
        error('Dane wejściowe zawierają znaki nie będące cyframi');
    end

    if length(pesel) ~= 11
        error('PESEL musi mieć dokładnie 11 cyfr');
    end

    digits = arrayfun(@(x) str2double(x), pesel);
    weights = [1 3 7 9 1 3 7 9 1 3 1];
    weightedSum = sum(digits .* weights);

    isValid = mod(weightedSum, 10) == 0;

    if isValid
        if mod(digits(10), 2) == 0
            disp('Podała Pani prawidłowy numer PESEL');
        else
            disp('Podał Pan prawidłowy numer PESEL');
        end
        return;
    else
        disp('Niepoprawny PESEL');
        return;
    end
end
```

Poniżej przykładowe wywołania funkcji

```
validate('19220315474')
```

```
Podał Pan prawidłowy numer PESEL
ans = logical
   1
```

```
validate('12345678903')
```

```
Podała Pani prawidłowy numer PESEL
ans = logical
   1
```

```
validate('19900101234')
```

```
Niepoprawny PESEL
ans = logical
   0
```

```
validate('199001012342')
```

```
Error using authenticate>validate (line 9)
PESEL musi mieć dokładnie 11 cyfr
```

```
validate('123abc78903')
```

```
Error using authenticate>validate (line 5)
Dane wejściowe zawierają znaki nie będące cyframi
```

Zadanie 2

Funkcja bada wzajemne położenie dwóch okręgów. Jako argumenty przyjmuje promienie **`R1`** i **`R2`** oraz środki okręgów **`center1`** i **`center2`** będące w formacie **`[x,y]`**.

```
function position = checkCirclePosition(R1, center1, R2, center2)
    if R1 <= 0 || R2 <= 0
        error('Promienie muszą być dodatnie');
    end

    distance = norm(center1 - center2);

    if distance == 0
        position = 'Okręgi są współśrodkowe';
    elseif distance > R1 + R2
        position = 'Okręgi są rozłączne zewnętrznie';
    elseif distance < abs(R1 - R2)
        position = 'Okręgi są rozłączne wewnętrznie';
    elseif distance == R1 + R2
        position = 'Okręgi są styczne zewnętrznie';
    elseif distance == abs(R1 - R2)
        position = 'Okręgi są styczne wewnętrznie';
    elseif abs(R1 - R2) < distance && distance < R1 + R2
        position = 'Okręgi przecinają się';
    end

    disp(position);
end
```

Poniżej przykładowe wywołania funkcji

```
checkCirclePosition(2, [-4,3], 3, [2,3]);
```

```
Okręgi są rozłączne zewnętrznie
```

```
checkCirclePosition(1, [1,4], 3, [2,3]);
```

```
Okręgi są rozłączne wewnętrznie
```

```
checkCirclePosition(3, [-4,3], 3, [2,3]);
```

```
Okręgi są styczne zewnętrznie
```

```
checkCirclePosition(2, [1,3], 3, [2,3]);
```

```
Okręgi są styczne wewnętrznie
```

```
checkCirclePosition(2, [-1,4], 3, [2,3]);
```

```
Okręgi przecinają się
```

```
checkCirclePosition(2, [2,3], 3, [2,3]);
```

```
Okręgi są współśrodkowe
```

Zadanie 3

Funkcja zwraca dla dodatniej liczby naturalnej $n \in \mathbb{N}_+$ wszystkie liczby pierwsze p w przedziale $p \in (0, n]$ korzystając z ***algorytmu sita Eratostenesa***.

```
function primes = sieveOfEratosthenes(n)
    if n <= 0
        error('Input must be a positive integer');
    end
    A = true(1, n);
    A(1) = false;

    for i = 2:sqrt(n)
        if A(i)
            A(i^2:i:n) = false;
        end
    end

    primes = find(A);
end
```

Poniżej przykładowe wywołania funkcji

```
sieveOfEratosthenes(10)
```

```
ans = 1×4
     2     3     5     7
```

```
sieveOfEratosthenes(50)
```

```
ans = 1×15
     2     3     5     7    11    13    17    19    23    29    31    37    41 ···
```

```
sieveOfEratosthenes(-10)
```

```
Error using authenticate>sieveOfEratosthenes (line 62)
Input must be a positive integer
```