1 Punto 2 B

```python
import numpy as np
from scipy.constants import m_e

k_BT = 0.02585

N_c = 8.86e18
N_v = 8.86e18

m_e_s = 0.5 * m_e
m_h_s = 0.5 * m_e

E_g_silicio = 1.1

E_g_germanio = 0.75

def n_i(E_g):
    return np.sqrt(N_c * N_v) * np.exp( - E_g / (2 * k_BT))

n_i_silicio = n_i(E_g_silicio)
n_i_germanio = n_i(E_g_germanio)
if __name__=="__main__":
    print(n_i_silicio)
    print(n_i_germanio)
```

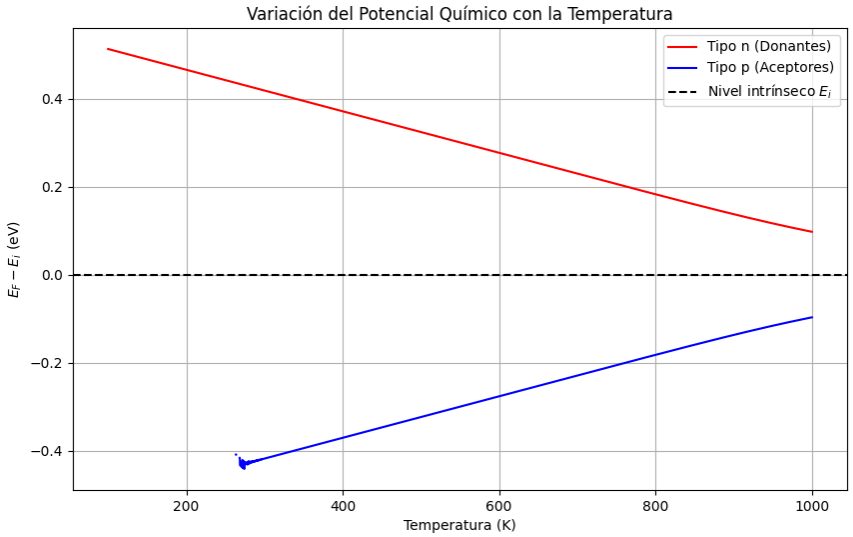
2 Punto 2 C

```python
from punto_2_b import n_i_germanio, n_i_silicio

N_max_silicio = 0.1 * n_i_silicio
N_max_germanio = 0.1 * n_i_germanio


print(N_max_silicio)
print(N_max_germanio)
```

3 Punto 3 B

```python
import numpy as np
from typing import Literal
import matplotlib.pyplot as plt

k = 8.617e-5
Eg = 1.12
N0 = 2.4e19

Nd = 1e17
Na = 1e17

T = np.linspace(100, 1000, 5000)

def var_don(N, T, type: Literal["p", "n"] = "n"):
    Ef = np.zeros_like(T)
    for i, temp in enumerate(T):
        A = (N/N0) * np.exp(Eg / (2 * k * temp))
        x = np.log(((1 if type == "n" else -1) * A + np.sqrt(A**2 + 4)) / 2)
```

```
        Ef[i] = k * temp * x
    return Ef

Ef_n = var_don(Nd, T)
Ef_p = var_don(Na, T, type="p")

plt.figure(figsize=(10, 6))
plt.plot(T, Ef_n, 'r-', label='Tipo n (Donantes)')
plt.plot(T, Ef_p, 'b-', label='Tipo p (Aceptores)')
plt.axhline(y=0, color='k', linestyle='--', label='Nivel intrinseco $E_i$')
plt.xlabel('Temperatura (K)')
plt.ylabel('$E_F - E_i$ (eV)')
plt.title('Variacion del Potencial Quimico con la Temperatura')
plt.legend()
plt.grid(True)
plt.savefig("punto3b.png")
```



4 Punto 3 C

```
import numpy as np
from numpy import pi
from scipy.constants import h, m_e, k

T = 300

def density(mc, mv, Eg):
    def cal_N(m):
        return 2 * np.pow((2 * pi * m * k * T)/h**2, (3/2))
    N_c = cal_N(mc)
    N_v = cal_N(mv)
    ex = np.exp(- (Eg) / (k * T))
    return N_c * N_v * ex

E = 1.602176634e-19

```

```
n_i_squared = density(0.25 * m_e, 0.4 * m_e, E)
n_i = np.sqrt(n_i_squared)
print("Densidad intrinseca n_i =", n_i, "m^-3")

n = 1e18  # m^-3
p = n_i_squared / n
print("Densidad de huecos p =", p, "m^-3")
```

5 Punto 5 C

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

J = -1.0
z = 6
kB = 1.0

T_N = abs(J) * z / (4 * kB)

def equation(x, T):
    if T == 0:
        return 0.5
    if T >= T_N:
        return 0.0
    return x - 0.5 * np.tanh(abs(J) * z * x / (2 * kB * T))

T_range = np.linspace(0.01, 2 * T_N, 200)
x_solutions = []

for T in T_range:
    if T >= T_N:
        x_solutions.append(0.0)
    else:
        initial_guesses = [0.1, 0.2, 0.3, 0.4, 0.5]
        found_solution = False

        for guess in initial_guesses:
            try:
                sol = fsolve(equation, guess, args=(T,), full_output=True)
                if sol[2] == 1 and sol[0][0] > 1e-6:
                    x_solutions.append(float(sol[0][0]))
                    found_solution = True
                    break
            except:
                continue

        if not found_solution:
            x_solutions.append(0.0)

x_solutions = np.array(x_solutions)

plt.figure(figsize=(10, 6))
plt.plot(T_range, x_solutions, 'b-', linewidth=3, label=r'$x = \frac{1}{2}\tanh\
    left(\frac{|J|z x}{2k_B T}\right)$')
plt.fill_between(T_range, 0, x_solutions, where=(T_range < T_N),
                 alpha=0.2, color='blue', label='Fase ordenada ($T < T_N$)')
plt.fill_between(T_range, 0, 0.001, where=(T_range >= T_N),
```

```
                alpha=0.2, color='red', label='Fase paramagnetica ($T <= T_N$)')
plt.xlabel('Temperatura (T)', fontsize=14)
plt.ylabel(r'$x = |s_A| = |s_B|$', fontsize=14)
plt.title('Solucion de la ecuacion autoconsistente: $x = \\frac{1}{2}\\tanh\\left
    (\\frac{|J|z x}{2k_B T}\\right)$',
          fontsize=16, fontweight='bold')
plt.xlim([0, 2 * T_N])
plt.ylim([-0.05, 0.55])
plt.grid(True, alpha=0.3)
plt.legend(loc='upper right', fontsize=12)
plt.tight_layout()
plt.savefig("./punto_5c.png")
```