In [3]:

```
import numpy as np
import math
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

Exercise 1

In [4]:

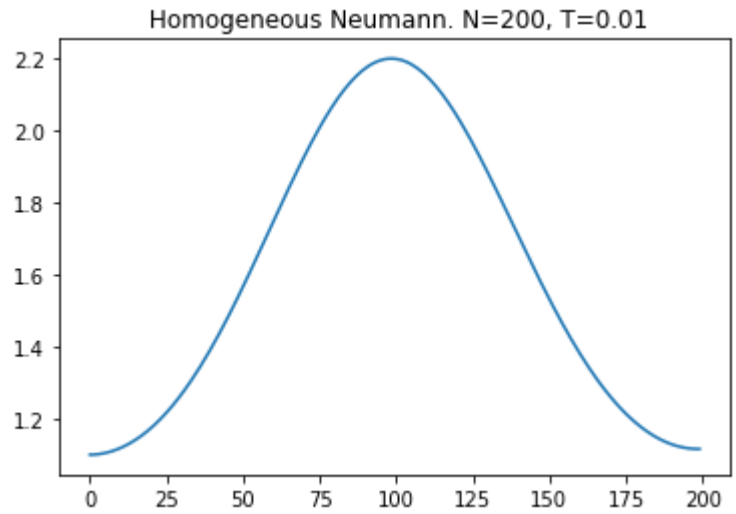
```
def set_u0(x):
    if 0.4 <= x < 0.6:
        return 4
    else:
        return 1
```

In [5]:

```
N = 200
T = 0.01

a = 1
b = 0
c = 1

h = (a - b) / N

dt = h*h / (2 * c) * 0.9

space = np.linspace(a, b, N)

u0 = np.array(list(map(set_u0, space)))
u1 = np.zeros(u0.shape)

boundary = 'Neumann'
```

In [6]:

```
t = 0
while t < T:
    for i in range(1, N - 1):
        u1[i] = u0[i] + c * (dt / (h*h)) * (u0[i-1] - 2*u0[i] + u0[i+1])
        if boundary == 'Dirichlet':
            u1[0] = u0[0]
            u1[N-1] = u0[N-1]
        elif boundary == 'Neumann':
            u1[0] = u0[1]
            u1[N - 1] = u0[N - 2]

    u0 = u1
    t += dt


plt.title(f" Homogeneous {boundary}. N={N}, T={T}")
plt.plot(u0)
plt.show()
```



Exercise 2

In [7]:

```
lam = 1
n = 10
B = np.zeros((n, n))

for i in range(n):
    for j in range(n):
        if i == 0 and j == n-1:
            B[i][j] = (-0.5)*lam
        elif i == n-1 and j == 0:
            B[i][j] = (-0.5)*lam
        elif i == j:
            B[i][j] = 1 + lam
        elif i == j-1:
            B[i][j] = (-0.5)*lam
        elif i-1 == j:
            B[i][j] = (-0.5)*lam

print("B:\n", B)

D = np.zeros((n, n))

for i in range(n):
    for j in range(n):
        if i == 0 and j == n-1:
            D[i][j] = (0.5)*lam
        elif i == n-1 and j == 0:
            D[i][j] = (0.5)*lam
        elif i == j:
            D[i][j] = 1 - lam
        elif i == j-1:
            D[i][j] = (0.5)*lam
        elif i-1 == j:
            D[i][j] = (0.5)*lam

print("D:\n", D)
```

```
B:
 [[ 2.  -0.5  0.   0.   0.   0.   0.   0.   0.  -0.5]
 [-0.5  2.  -0.5  0.   0.   0.   0.   0.   0.   0. ]
 [ 0.  -0.5  2.  -0.5  0.   0.   0.   0.   0.   0. ]
 [ 0.   0.  -0.5  2.  -0.5  0.   0.   0.   0.   0. ]
 [ 0.   0.   0.  -0.5  2.  -0.5  0.   0.   0.   0. ]
 [ 0.   0.   0.   0.  -0.5  2.  -0.5  0.   0.   0. ]
 [ 0.   0.   0.   0.   0.  -0.5  2.  -0.5  0.   0. ]
 [ 0.   0.   0.   0.   0.   0.  -0.5  2.  -0.5  0. ]
 [ 0.   0.   0.   0.   0.   0.   0.  -0.5  2.  -0.5]
 [-0.5  0.   0.   0.   0.   0.   0.   0.  -0.5  2. ]]
D:
 [[0.  0.5 0.  0.  0.  0.  0.  0.  0.  0.5]
 [0.5 0.  0.5 0.  0.  0.  0.  0.  0.  0. ]
 [0.  0.5 0.  0.5 0.  0.  0.  0.  0.  0. ]
 [0.  0.  0.5 0.  0.5 0.  0.  0.  0.  0. ]
 [0.  0.  0.  0.5 0.  0.5 0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.5 0.  0.5 0.  0.  0. ]
 [0.  0.  0.  0.  0.  0.5 0.  0.5 0.  0. ]
 [0.  0.  0.  0.  0.  0.  0.5 0.  0.5 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.5 0.  0.5]
 [0.5 0.  0.  0.  0.  0.  0.  0.  0.5 0. ]]
```

In [8]:

```
u1 = np.eye(n)
u = np.eye(n)

x = B @ u1
y = D @ u

print("x:\n", x)
print("y:\n", y)

ans = np.linalg.solve(x, y)
print("ans:\n", ans)
```

```
  0.00637959 0.02232855 0.08293461 0.30940989]
 [0.30940989 0.15470494 0.30940989 0.08293461 0.02232855 0.00637959
  0.00318979 0.00637959 0.02232855 0.08293461]
 [0.08293461 0.30940989 0.15470494 0.30940989 0.08293461 0.02232855
  0.00637959 0.00318979 0.00637959 0.02232855]
 [0.02232855 0.08293461 0.30940989 0.15470494 0.30940989 0.08293461
  0.02232855 0.00637959 0.00318979 0.00637959]
 [0.00637959 0.02232855 0.08293461 0.30940989 0.15470494 0.30940989
  0.08293461 0.02232855 0.00637959 0.00318979]
 [0.00318979 0.00637959 0.02232855 0.08293461 0.30940989 0.15470494
  0.30940989 0.08293461 0.02232855 0.00637959]
 [0.00637959 0.00318979 0.00637959 0.02232855 0.08293461 0.30940989
  0.15470494 0.30940989 0.08293461 0.02232855]
 [0.02232855 0.00637959 0.00318979 0.00637959 0.02232855 0.08293461
  0.30940989 0.15470494 0.30940989 0.08293461]
 [0.08293461 0.02232855 0.00637959 0.00318979 0.00637959 0.02232855
  0.08293461 0.30940989 0.15470494 0.30940989]
 [0.30940989 0.08293461 0.02232855 0.00637959 0.00318979 0.00637959
  0.02232855 0.08293461 0.30940989 0.15470494]]
```

Exercise 3

In [9]:

```
n = 200
delta_x = 2 / n
x = [0] * n
for i in range(n):
    x[i] = -1 + i * delta_x

print("x")
print(x)
```

```
x
[-1.0, -0.99, -0.98, -0.97, -0.96, -0.95, -0.94, -0.9299999999999999,
-0.92, -0.91, -0.9, -0.89, -0.88, -0.87, -0.86, -0.85, -0.84, -0.83, -
0.8200000000000001, -0.81, -0.8, -0.79, -0.78, -0.77, -0.76, -0.75, -
0.74, -0.73, -0.72, -0.71, -0.7, -0.69, -0.6799999999999999, -0.669999
9999999999, -0.6599999999999999, -0.6499999999999999, -0.64, -0.63, -
0.62, -0.61, -0.6, -0.59, -0.5800000000000001, -0.5700000000000001, -
0.56, -0.55, -0.54, -0.53, -0.52, -0.51, -0.5, -0.49, -0.48, -0.47, -
0.45999999999999996, -0.44999999999999996, -0.43999999999999995, -0.42
999999999999994, -0.42000000000000004, -0.41000000000000003, -0.4, -0.
39, -0.38, -0.37, -0.36, -0.35, -0.33999999999999997, -0.3299999999999
9996, -0.31999999999999995, -0.30999999999999994, -0.2999999999999999
3, -0.29000000000000004, -0.28, -0.27, -0.26, -0.25, -0.24, -0.2299999
9999999998, -0.21999999999999997, -0.20999999999999996, -0.19999999999
999996, -0.18999999999999995, -0.17999999999999994, -0.169999999999999
93, -0.16000000000000003, -0.15000000000000002, -0.14, -0.13, -0.12, -
0.10999999999999999, -0.09999999999999998, -0.08999999999999997, -0.07
999999999999996, -0.06999999999999995, -0.05999999999999994, -0.049999
99999999993, -0.040000000000000036, -0.030000000000000027, -0.02000000
0000000018, -0.010000000000000009, 0.0, 0.010000000000000009, 0.020000
000000000018, 0.030000000000000027, 0.040000000000000036, 0.0500000000
00000044, 0.06000000000000005, 0.07000000000000006, 0.0800000000000000
7, 0.09000000000000008, 0.10000000000000009, 0.11000000000000001, 0.120
0000000000001, 0.13000000000000012, 0.14000000000000012, 0.15000000000
000013, 0.15999999999999992, 0.16999999999999993, 0.17999999999999994,
0.18999999999999995, 0.19999999999999996, 0.20999999999999996, 0.21999
999999999997, 0.22999999999999998, 0.24, 0.25, 0.26, 0.27, 0.28, 0.290
00000000000004, 0.30000000000000004, 0.31000000000000005, 0.3200000000
0000006, 0.33000000000000007, 0.3400000000000001, 0.3500000000000001,
0.3600000000000001, 0.3700000000000001, 0.3800000000000001, 0.39000000
00000001, 0.40000000000000013, 0.4099999999999999, 0.4199999999999999
3, 0.42999999999999994, 0.43999999999999995, 0.44999999999999996, 0.45
999999999999996, 0.47, 0.48, 0.49, 0.5, 0.51, 0.52, 0.53, 0.54, 0.55,
0.56, 0.5700000000000001, 0.5800000000000001, 0.5900000000000001, 0.60
00000000000001, 0.6100000000000001, 0.6200000000000001, 0.630000000000
0001, 0.6400000000000001, 0.6500000000000001, 0.6600000000000001, 0.66
99999999999999, 0.6799999999999999, 0.69, 0.7, 0.71, 0.72, 0.73, 0.74,
0.75, 0.76, 0.77, 0.78, 0.79, 0.8, 0.81, 0.8200000000000001, 0.8300000
000000001, 0.8400000000000001, 0.8500000000000001, 0.8600000000000001,
0.8700000000000001, 0.8800000000000001, 0.8900000000000001, 0.90000000
00000001, 0.9100000000000001, 0.9199999999999999, 0.9299999999999999,
0.94, 0.95, 0.96, 0.97, 0.98, 0.99]
```

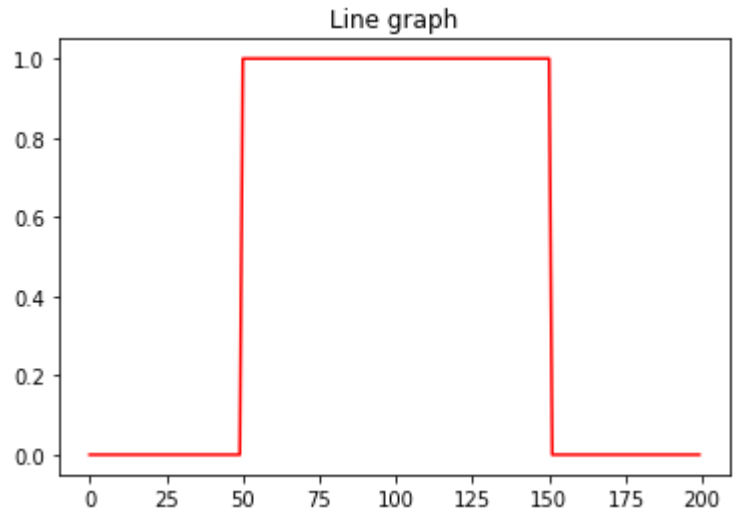
In [10]:

```
phi_u = [0] * n
for i in range(n):
    if abs(x[i]) <= 0.5:
        phi_u[i] = 1

print("phi_u:")
print(phi_u)
```

```
phi_u:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [11]:

```
plt.title("Line graph")
plt.plot(phi_u, color="red")

plt.show()
```



In [28]:

```
CFL = 0.8
c = 1
delta_t = delta_x * CFL
time = 100
method = 0.1

phi_u = [0] * n
for i in range(n):
    if abs(x[i]) <= 0.5:
        phi_u[i] = 1

for t in range(time):
    new_u = list(phi_u)
    for i in range(0, n):
        new_u[i] = phi_u[i-1] - c * (delta_t / delta_x) * (method)
    phi_u = new_u
    plt.title("Line graph")
    plt.plot(new_u, color="red")
    plt.show()
```