This is a simulation of the sun

Import the required package

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [11]:
```
#Define the constants

sig = 5.670367e-8
# Stefan-Boltzmann constant in W/m^2*K^4

a = 7.565767e-16
# Radiation constant in J/m^3*K^4

k_B = 1.38064852e-23
# Boltzmann constant in J/K

mu = 1.18
# Mean molecular weight for the Sun (dimensionless)

m_H = 1.6735e-27
# Hydrogen atom mass in kg

G = 6.674e-11
# Gravitational constant in m^3/kg*s^2

const = k_B / (mu * m_H)
# Constant used in the ideal gas law

Rsun = 696300e3
# Solar radius in meters

Msun = 1.989e30
# Solar mass in kg

Lsun = 3.846e26
# Solar luminosity in Watts

kappa_Th = 0.04e-4 / 1e-3
# Opacity constant for Thompson scattering

eps_0 = Lsun / (0.15 * Msun)
# Central energy production rate

coeff = 600
# Coefficient used in opacity calculation
```

set up the initial conditions

In [12]:
```
L = 0.  # Initial luminosity

m = 0.  # Initial mass

T0 = 17e6  # Initial temperature

rho0 = 162e3  # Initial density

T = T0  # Current temperature

rho = rho0  # Current density

pr = a * T**4 / 3  # Radiation pressure

pg = const * T * rho  # Gas pressure

p = pr + pg  # Total pressure

p0 = p  # Initial pressure

kappa = kappa_Th  # Initial opacity

N = 300  # Number of radial shells for modeling the Sun

dr = Rsun / N  # Shell thickness
```

Store the results

In [13]:
```
struct = np.zeros((14, N+1))
struct[0, 0] = 0.
# Initial radius
# (other quantities are initialized to zero)

print(f'T0 = {T0 / 1e6} mln K')
print(f'rho0 = {rho0 / 1e3} g/cm^3')
```

```
T0 = 17.0 mln K
rho0 = 162.0 g/cm^3
```

In [14]:
```
## Perform calculation from inside to outside
for i in range(1, N+1):

    r = dr * (i - 0.5)
    # Position of the current shell
    surface = 4 * np.pi * r**2
    # Surface area of the shell
    dvol = surface * dr
    # Volume of the shell
    pg = const * T * rho
    # Update the gas pressure
    dm = dvol * rho
    # Mass of the shell
    m += dm
    # Accumulate the total mass
    ratio = (1. - L/Lsun)
    if ratio < 0:
            eps = 0.
    else:
        eps = eps_0 * max(ratio**0.75, 0.)

    # Energy production rate
    L += eps * dm
    # Update the luminosity
    kappa = kappa_Th * (1. + coeff * (r/Rsun)**1.15 + 60 * (r/Rsun)**4.0)
    # Opacity calculation
    dTdr = -L * 3 * kappa * rho / (16 * sig * T**3 * surface)
    # Temperature gradient
    T = max(T + dTdr * dr, 5780.)
    # Update the temperature
    pr = a * T**4 / 3
    # Update the radiation pressure
    dPdr = -G * m / r**2 * rho
    # Pressure gradient
    p += dPdr * dr
    # Update the pressure
    pg = p - pr
    # Compute the new gas pressure
    rho = pg / (const * T)
    # Update the density

    # Check for physical validity of the computed values
    if p <= 0 or pr <= 0 or pg <= 0:
        break

    # Store the results in the data structure
    struct[0, i] = r
    struct[1, i] = m
    struct[3, i] = T
    struct[4, i] = p
    struct[5, i] = pr
    struct[6, i] = pg
    struct[7, i] = rho
    struct[10, i] = L
    struct[11, i] = L / (4 * np.pi * r**2)
    struct[12, i] = eps
```

```
    struct[13, i] = kappa
```

Plotting the results

In [15]:
```
x = struct[0, :] / Rsun  # Normalize radius by solar radius for plotting

plt.figure(figsize=(10, 6))
plt.plot(x, struct[7, :] / rho0, label="rho(r)")
# Density
plt.plot(x, struct[1, :] / Msun, label="M(r)")
# Mass
plt.plot(x, struct[3, :] / T0, label="T(r)", linestyle="--")
# Temperature
plt.plot(x, struct[4, :] / p0, label="P(r)", linestyle=":")
# Pressure
plt.plot(x, struct[10, :] / Lsun, label="L(r)", linestyle="-.")
# Luminosity
plt.xlabel("r/R")
plt.ylabel("Normalized values")
plt.title("Solar model with T(0)=17 mln K, rho(0)=162 g/cm^3")
plt.legend()
plt.grid(True)
plt.show()
```