```python
# Import modules

import matplotlib.pyplot as plt
import numpy as np
from q2_b import g_star, g_star_s

# Define a, t ranges
a_range = np.arange(np.log(1e11), np.log(1e19), 0.0075)
t_range = np.arange(np.log(1e12), np.log(1e26), 0.0075)

# Define Constants
T_rh = 1e14
M_pl = 2.4e18
g_high = g_star(T_rh)
t_rh = ((3 * M_pl * np.sqrt(10)) / (2 * np.pi)) * 1 / (T_rh ** 2 * np.sqrt(g_high
))

# Define T thresholds
T_thres = [100, 30, 15, 1, 0.2, 0.05, 0.00025]  # GeV

# Define a thresholds
a_thres = [T_rh / T_crit for T_crit in T_thres]

# Data structure to help later
g_before_after = []
for T in T_thres:
    g_before_after.append([g_star_s(T * 1000 + 10), g_star_s(T * 1000 - 10)])
g_star_before_after = []
for T in T_thres:
    g_star_before_after.append([g_star(T * 1000 + 10), g_star(T * 1000 - 10)])


def T_approx(a):
    T_a_prop_const = T_rh * g_high ** (1 / 3)
    if np.exp(a) < a_thres[0]:
        return T_a_prop_const * (g_before_after[0][0] ** (-1 / 3)) / np.exp(a)
    elif a_thres[0] ⩽ np.exp(a) < a_thres[1]:
        return T_a_prop_const * (g_before_after[0][1] ** (-1 / 3)) / np.exp(a)
    elif a_thres[1] ⩽ np.exp(a) < a_thres[2]:
        return T_a_prop_const * (g_before_after[1][1] ** (-1 / 3)) / np.exp(a)
    elif a_thres[2] ⩽ np.exp(a) < a_thres[3]:
        return T_a_prop_const * (g_before_after[2][1] ** (-1 / 3)) / np.exp(a)
    elif a_thres[3] ⩽ np.exp(a) < a_thres[4]:
        return T_a_prop_const * (g_before_after[3][1] ** (-1 / 3)) / np.exp(a)
    elif a_thres[4] ⩽ np.exp(a) < a_thres[5]:
        return T_a_prop_const * (g_before_after[4][1] ** (-1 / 3)) / np.exp(a)
    elif a_thres[5] ⩽ np.exp(a) < a_thres[6]:
        return T_a_prop_const * (g_before_after[5][1] ** (-1 / 3)) / np.exp(a)
    elif a_thres[6] ⩽ np.exp(a):
        return T_a_prop_const * (g_before_after[6][1] ** (-1 / 3)) / np.exp(a)


def T_approx_exp(a):
    T_a_prop_const = T_rh * g_high ** (1 / 3)
    if a < a_thres[0]:
        return T_a_prop_const * (g_before_after[0][0] ** (-1 / 3)) / a
    elif a_thres[0] ⩽ a < a_thres[1]:
```

```
57          return T_a_prop_const * (g_before_after[0][1] ** (-1 / 3)) / a
58      elif a_thres[1] ⩽ a < a_thres[2]:
59          return T_a_prop_const * (g_before_after[1][1] ** (-1 / 3)) / a
60      elif a_thres[2] ⩽ a < a_thres[3]:
61          return T_a_prop_const * (g_before_after[2][1] ** (-1 / 3)) / a
62      elif a_thres[3] ⩽ a < a_thres[4]:
63          return T_a_prop_const * (g_before_after[3][1] ** (-1 / 3)) / a
64      elif a_thres[4] ⩽ a < a_thres[5]:
65          return T_a_prop_const * (g_before_after[4][1] ** (-1 / 3)) / a
66      elif a_thres[5] ⩽ a < a_thres[6]:
67          return T_a_prop_const * (g_before_after[5][1] ** (-1 / 3)) / a
68      elif a_thres[6] ⩽ a:
69          return T_a_prop_const * (g_before_after[6][1] ** (-1 / 3)) / a
70
71
72  # Plot T(a)
73  T_approx_array = [T_approx(a_val) for a_val in a_range]
74  plt.loglog(np.exp(a_range), T_approx_array, 'b-', label=r'Piecewise Approx.',
    linewidth=1)
75  for a in a_thres:
76      plt.vlines(a, min(T_approx_array), max(T_approx_array), linestyles='dashed',
    colors='k', linewidth=0.5)
77  plt.title(r'$T(a)$ Transition')
78  plt.ylabel(r'$T$ (GeV)')
79  plt.xlabel(r'$a$')
80  plt.legend(loc='best')
81  plt.savefig('T_a_transition_universe', dpi=300)
82  plt.show()
83
84
85  # Calculate t0 depending on the thresholds
86  def calculatet0(T_thres, i):
87      return ((3 * M_pl * np.sqrt(10)) / (2 * np.pi)) * 1 / (
88              T_thres ** 2 * np.sqrt(0.5 * (g_before_after[i][0] + g_before_after[i
    ][1])))
89
90
91  t0 = []
92  for i in range(len(T_thres)):
93      t0.append(calculatet0(T_thres[i], i))
94
95  a_t_prop_const = (1 / np.sqrt(t_rh)) * np.sqrt(g_star(T_rh) ** (1 / 6))
96
97
98  def a_piecewise(t):
99      a_t_prop_const = (1 / np.sqrt(t_rh)) * np.sqrt(g_star(T_rh) ** (1 / 6))
100     if np.exp(t) < t0[0]:
101         return a_t_prop_const * np.sqrt(
102             (g_star_before_after[0][0] ** (1 / 2)) / (g_before_after[0][0] ** (2
     / 3))) * np.sqrt(
103             np.exp(t))
104     elif t0[0] ⩽ np.exp(t) < t0[1]:
105         return a_t_prop_const * np.sqrt(
106             (g_star_before_after[0][1] ** (1 / 2)) / (g_before_after[0][1] ** (2
     / 3))) * np.sqrt(
107             np.exp(t))
108     elif t0[1] ⩽ np.exp(t) < t0[2]:
```

```
        return a_t_prop_const * np.sqrt(
            (g_star_before_after[1][1] ** (1 / 2)) / (g_before_after[1][1] ** (2 / 3))) * np.sqrt(
            np.exp(t))
    elif t0[2] ⩽ np.exp(t) < t0[3]:
        return a_t_prop_const * np.sqrt(
            (g_star_before_after[2][1] ** (1 / 2)) / (g_before_after[2][1] ** (2 / 3))) * np.sqrt(
            np.exp(t))
    elif t0[3] ⩽ np.exp(t) < t0[4]:
        return a_t_prop_const * np.sqrt(
            (g_star_before_after[3][1] ** (1 / 2)) / (g_before_after[3][1] ** (2 / 3))) * np.sqrt(
            np.exp(t))
    elif t0[4] ⩽ np.exp(t) < t0[5]:
        return a_t_prop_const * np.sqrt(
            (g_star_before_after[4][1] ** (1 / 2)) / (g_before_after[4][1] ** (2 / 3))) * np.sqrt(
            np.exp(t))
    elif t0[5] ⩽ np.exp(t) < t0[6]:
        return a_t_prop_const * np.sqrt(
            (g_star_before_after[5][1] ** (1 / 2)) / (g_before_after[5][1] ** (2 / 3))) * np.sqrt(
            np.exp(t))
    elif t0[6] ⩽ np.exp(t):
        return a_t_prop_const * np.sqrt(
            (g_star_before_after[6][1] ** (1 / 2)) / (g_before_after[6][1] ** (2 / 3))) * np.sqrt(
            np.exp(t))


# Plot a(t)
a_approx_array = [a_piecewise(t_val) for t_val in t_range]
plt.loglog(np.exp(t_range), a_approx_array, 'b-', label=r'Piecewise Approx.', linewidth=1)
for t0_val in t0:
    plt.vlines(t0_val, min(a_approx_array), max(a_approx_array), linestyles='dashed', colors='k', linewidth=0.5)
plt.title(r'$a(t)$ Transition')
plt.ylabel(r'$a$')
plt.xlabel(r'$t$ (1/GeV)')
plt.legend(loc='best')
plt.savefig('a_t_transition_universe', dpi=300)
plt.show()

# Plot T(t)
T_t_array = [T_approx_exp(a_val) for a_val in a_approx_array]
plt.loglog(np.exp(t_range), T_t_array, 'b-', label=r'Piecewise Approx.', linewidth=1)
for t0_val in t0:
    plt.vlines(t0_val, min(T_t_array), max(T_t_array), linestyles='dashed', colors='k', linewidth=0.5)
plt.title(r'$T(t)$ Transition')
plt.ylabel(r'$T$ (GeV)')
plt.xlabel(r'$t$ (1/Gev)')
plt.legend(loc='best')
plt.savefig('T_t_transition_universe', dpi=300)
```

```
plt.show()


# Print
def latex_float(f):
    float_str = "{0:.3g}".format(f)
    if "e" in float_str:
        base, exponent = float_str.split("e")
        return r"{0} \times 10^{{{1}}}".format(base, int(exponent))
    else:
        return float_str


t_print = [i * np.log(10) for i in range(13, 27)]
print("a(t), T(t)")
for i in range(len(t_print)):
    print(
        "$10^{{{}}}$ & ${}$ & {} \\\\ \hline".format(i + 13, latex_float(a_piecewise(t_print[i])),
                                                     latex_float(T_approx_exp(a_piecewise(t_print[i])))))

```