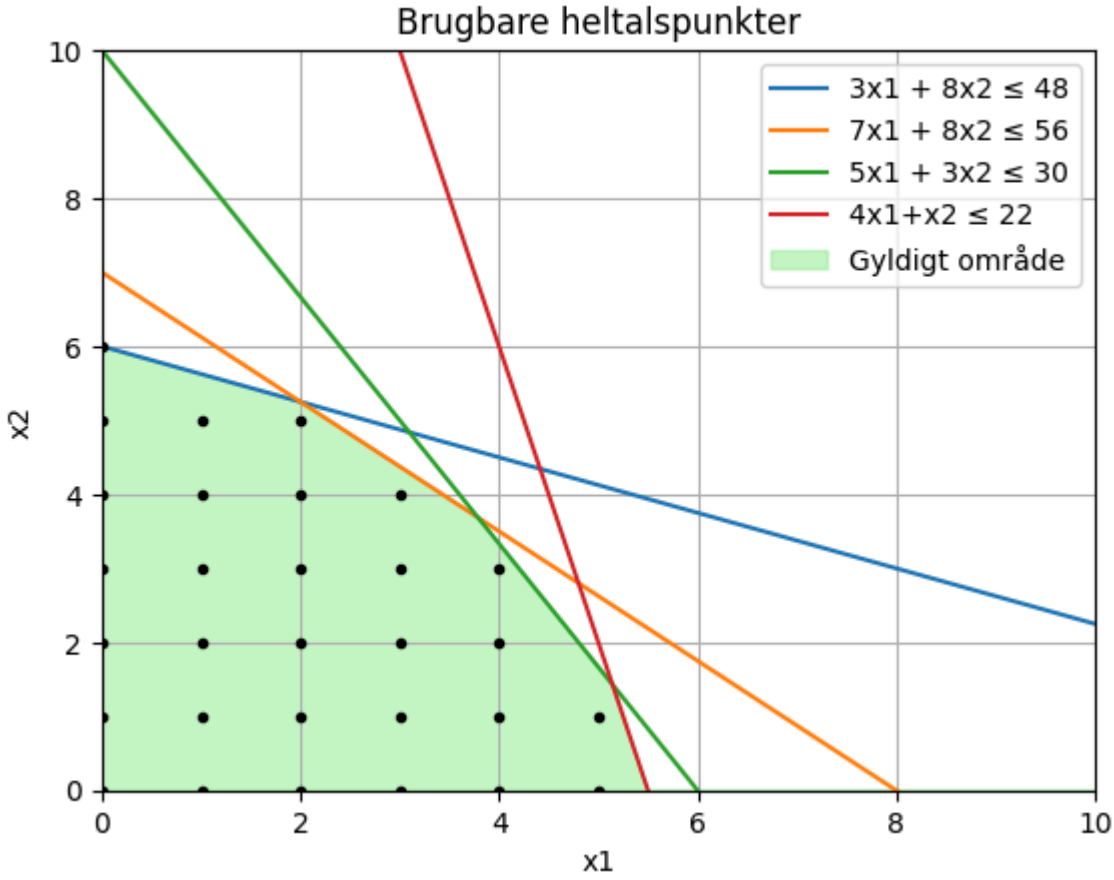
Eksamen 2021, opgave 2

Spm. 1) Marker de brugbare heltalspunkter i en grafisk fremstilling som i Spm. 1) i Opgave 1.

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

# Definer uligheder som funktioner
def f1(x): return (48 - 3*x)/8
def f2(x): return (56 - 7*x)/8
def f3(x): return (30 - 5*x)/3
def f4(x): return 22 - 4*x

x = np.linspace(0, 10, 400)
y1 = f1(x)
y2 = f2(x)
y3 = f3(x)
y4 = f4(x)

# Plot begrænsninger
plt.plot(x, y1, label='3x1 + 8x2 ≤ 48')
plt.plot(x, y2, label='7x1 + 8x2 ≤ 56')
plt.plot(x, y3, label='5x1 + 3x2 ≤ 30')
plt.plot(x, y4, label='4x1+x2 ≤ 22')

# Udfyld det gyldige område
x_fill = np.linspace(0, 10, 400)
y_fill = np.minimum(np.minimum(f1(x_fill), f2(x_fill)),
                    np.minimum(f3(x_fill), f4(x_fill)))
plt.fill_between(x_fill, y_fill, color='lightgreen', alpha=0.5,
                 label='Gyldigt område')

# Plot heltallige løsninger i området
for x1 in range(11):
    for x2 in range(11):
        if (
            (3*x1 + 8*x2 <= 48) and
            (7*x1 + 8*x2 <= 56) and
            (5*x1 + 3*x2 <= 30) and
            (4*x1 + x2 <= 22)
        ):
            plt.plot(x1, x2, 'ko', markersize=3)

plt.xlim(0, 10)
plt.ylim(0, 10)
plt.xlabel('x1')
plt.ylabel('x2')
plt.grid(True)
plt.legend()
plt.title('Brugbare heltalspunkter')
```

Out[ ]:
```
Text(0.5, 1.0, 'Brugbare heltalspunkter')
```



Spm. 2) Branch and bound kan anvendes til at løse et ILP problem (Integer Linear Programming problem), hvor heltalskravene relakseres, og hvor der branches på en enkelt beslutningsvariabel. Forklar metoden generelt og anvend metoden til løsning af det specifikke, givne ILP problem. Vis i den forbindelse det branch and bound træ, der konstrueres i løsningsprocessen. Du er velkommen til at benytte software til at løse de LP problemer, der opstår i knudepunkterne i branch and bound træet.

Branch and Bound er en metode til at løse heltalsprogrammeringsproblemer (ILP), hvor man bruger en systematisk opdeling af problemet i mindre underproblemer og udelukker dem, der ikke kan føre til bedre løsninger.

1. Relaksation: Løs problemet som et almindeligt LP-problem uden heltalskrav.
2. Branching: Vælg en variabel med ikke-heltal værdi og lav to nye delproblemer med mulige heltalsløsninger, hvor variablen enten
   - er mindre end/lig med det største heltal mindre end værdien.
   - er større end/lig med det mindste heltal større end værdien.
3. Bounding: Hvert underproblem løses som et LP-problem. Hvis løsningen:
   - er infeasible udelukkes knuden.
   - er heltallig gemmes løsningen som kandidat.
   - ikke er heltallig, men bedre end den bedste fundne løsning branches videre.
   - ikke er bedre end bedste fundne løsning udelukkes knuden.

In [13]:
```
import pulp as PLP

def branch_and_bound(name, extra_constraints, x1, x2):
    model = PLP.LpProblem(name, PLP.LpMaximize)
```

```
    # Objektfunktion
    model += 8*x1 + 7*x2

    # Begrænsninger
    model += 3*x1 + 8*x2 <= 48
    model += 7*x1 + 8*x2 <= 56
    model += 5*x1 + 3*x2 <= 30
    model += 4*x1 + x2 <= 22

    # Knudespecifikke begrænsninger
    for constraint in extra_constraints:
        model += constraint

    model.solve()

    print(f"{name}")
    print(f"Status: {PLP.LpStatus[model.status]}")
    if model.status == 1:
        print(f"x1 = {x1.varValue:.2f}, x2 = {x2.varValue:.2f}")
        print(f"Objektværdi = {PLP.value(model.objective):.2f}")
    else:
        print("Ingen løsning (infeasible)")

    print()

x1 = PLP.LpVariable("x1", lowBound=0, cat=PLP.LpContinuous)
x2 = PLP.LpVariable("x2", lowBound=0, cat=PLP.LpContinuous)

branch_and_bound("Rodknude", [], x1, x2)
```

```
Rodknude
Status: Optimal
x1 = 3.79, x2 = 3.68
Objektværdi = 56.11
```

Jeg brancher på x2, da x2 er længst fra et heltal og dermed mest kritisk for at opnå en heltallig løsning.

In [14]:
```
branch_and_bound("Knude 1: x2 ≤ 3", [x2 <= 3], x1, x2)
branch_and_bound("Knude 2: x1 ≤ 4", [x2 <= 3, x1 <= 4], x1, x2)
```

```
Knude 1: x2 ≤ 3
Status: Optimal
x1 = 4.20, x2 = 3.00
Objektværdi = 54.60

Knude 2: x1 ≤ 4
Status: Optimal
x1 = 4.00, x2 = 3.00
Objektværdi = 53.00
```

Løsningen er gyldig og x1 og x2 er heltallige. Jeg undersøger om der findes en bedre løsning af den anden gren fra knude 1:

In [15]:
```
branch_and_bound("Knude 3: x1 ≥ 5", [x2 <= 3, x1 >= 5], x1, x2)
```

```
Knude 3: x1 ≥ 5
Status: Optimal
x1 = 5.00, x2 = 1.67
Objektværdi = 51.67
```

Da objektværdien er mindre end for knude 2, branches ikke videre. Jeg undersøger om der findes en bedre løsning af den anden gren fra roden:

In [16]:
```
branch_and_bound("Knude 4: x2 ≥ 4", [x2 >= 4], x1, x2)
branch_and_bound("Knude 5: x1 ≤ 3", [x2 >= 4, x1 <= 3], x1, x2)
branch_and_bound("Knude 6: x2 ≤ 4", [x2 >= 4, x1 <= 3, x2 <= 4], x1, x2)
```

```
Knude 4: x2 ≥ 4
Status: Optimal
x1 = 3.43, x2 = 4.00
Objektværdi = 55.43

Knude 5: x1 ≤ 3
Status: Optimal
x1 = 3.00, x2 = 4.38
Objektværdi = 54.62

Knude 6: x2 ≤ 4
Status: Optimal
x1 = 3.00, x2 = 4.00
Objektværdi = 52.00
```

Løsningen er gyldig og x1 og x2 er heltallige men da objektværdien er mindre end for knude 2 er knude 2 fortsat den foreløbige optimale værdi. Jeg undersøger om der findes en bedre løsning af den anden gren fra knude 5:

In [17]:
```
branch_and_bound("Knude 7: x2 ≥ 5", [x2 >= 4, x1 <= 3, x2 >= 5], x1, x2)
branch_and_bound("Knude 8: x1 ≤ 2", [x2 >= 4, x1 <= 3, x2 >= 5, x1 <= 2], x1, x2
```

```
Knude 7: x2 ≥ 5
Status: Optimal
x1 = 2.29, x2 = 5.00
Objektværdi = 53.29

Knude 8: x1 ≤ 2
Status: Optimal
x1 = 2.00, x2 = 5.25
Objektværdi = 52.75
```

Da objektværdien er mindre end for knude 2, branches ikke videre. Jeg undersøger om der findes en bedre løsning af den anden gren fra knude 7:

In [18]:
```
branch_and_bound("Knude 9: x1 ≥ 3", [x2 >= 4, x1 <= 3, x2 >= 5, x1 >= 3], x1, x2
```

```
Knude 9: x1 ≥ 3
Status: Infeasible
Ingen løsning (infeasible)
```

Problemet er infeasible. Jeg undersøger om der findes en bedre løsning af den anden gren fra knude 4:

In [19]:
```
branch_and_bound("Knude 10: x1 ≥ 4", [x2 >= 4, x1 >= 4], x1, x2)
```

```
Knude 10: x1 ≥ 4
Status: Infeasible
Ingen løsning (infeasible)


Status: Infeasible
Ingen løsning (infeasible)
```

Problemet er infeasible.

Dermed viser knude 2 den optimale løsning på optimeringsproblemet, når x1 og x2 er heltallige:

In [20]:
```
branch_and_bound("Knude 2: x1 ≤ 4", [x2 <= 3, x1 <= 4], x1, x2)
```

```
Knude 2: x1 ≤ 4
Status: Optimal
x1 = 4.00, x2 = 3.00
Objektværdi = 53.00
```

Nedenfor ses det branch and bound træ, der konstrueres i løsningsprocessen:

In [21]:
```
def draw_tree():
    print("                  ROD")
    print("               /       \\")
    print("            /             \\")
    print("         /                   \\")
    print("      4:x2>=4               1:x2<=3")
    print("    /        \\            /         \\")
    print("10:x1>=4    5:x1<=3    3:x1>=5       2:x1<=4")
    print("           /        \\")
    print("        7:x2>=5   6:x2<=4")
    print("        /    \\")
    print("   9:x1>=3  8:x1<=2")

draw_tree()
```

```
                  ROD
               /       \
            /             \
         /                   \
      4:x2>=4               1:x2<=3
    /        \            /         \
10:x1>=4    5:x1<=3    3:x1>=5       2:x1<=4
           /        \
        7:x2>=5   6:x2<=4
        /    \
   9:x1>=3  8:x1<=2
```

Spm. 3) Opstil også en formulering vha. Python og PuLP af dette ILP problem, og løs den opstillede Python/PuLP heltalsmodel. Der er ikke krav om nogen bestemt Solver, der skal benyttes til løsning af denne model.

In [22]:

```python
import pulp as PLP

Model = PLP.LpProblem("Branch and Bound", PLP.LpMaximize)

x1 = PLP.LpVariable("x1",lowBound=0,cat=PLP.LpInteger) # Heltallige variabler
x2 = PLP.LpVariable("x2",lowBound=0,cat=PLP.LpInteger)

# Objektfunktion
Model += 8*x1 + 7*x2, "Objektfunktion"

# Begrænsninger
Model += 3*x1 + 8*x2 <= 48, "Begrænsning 1"
Model += 7*x1 + 8*x2 <= 56, "Begrænsning 2"
Model += 5*x1 + 3*x2 <= 30, "Begrænsning 3"
Model += 4*x1 + x2 <= 22, "Begrænsning 4"

Model.solve()

# Udskriv resultatet
print(f"Status: {PLP.LpStatus[Model.status]}")
print(f"x1 = {x1.varValue}")
print(f"x2 = {x2.varValue}")
print(f"Optimal værdi = {PLP.value(Model.objective)}")
```

```
Status: Optimal
x1 = 4.0
x2 = 3.0
Optimal værdi = 53.0
```