Anexo

Punto 5

In [ ]:
```
x_a1 = 13/2
x_a2 = 65/8

x_b1 = 19/2
x_b2 = 95/8

w_a1 = 10
w_a2 = 5

w_b1 = 6
w_b2 = 15
```

In [ ]:
```
#walrasian allocation for test

# x_a1 = 7
# x_a2 = 8.75

# x_b1 = 9
# x_b2 = 11.25
```

In [ ]:
```
def requirement(r,t):

    """
    # 0 < t < r is a requirement, otherwise the function will not work
    """

    if t > 0:
        if t < r:
            return True
        else:
            return False
    else:
        return False
```

In [ ]:
```
def g_allocation(x_1,x_2,w_1,w_2,r,t):


  if requirement(r,t) == False:
      raise Exception("The requirement is not satisfied")

  # g_a = (g_a^1,g_a^2)

  g_a1 = ((t/r)*w_1) + ((1-(t/r))*x_1)

  g_a2 = (t/r)*w_2 + (1-(t/r))*x_2

  # return g_a intro a list
  return (g_a1,g_a2)
```

In [ ]:
```
def feasibility(x_1a,x_2a,x_1b,x_2b,r,t,w_1a,w_2a,w_1b,w_2b):
    # 0 < t < r is a requirement, otherwise the function will not work

    if requirement(r,t) == False:
      raise Exception("The requirement is not satisfied")

    x_1 = r*x_1a + (r-t)*x_1b
    x_2 = r*x_2a + (r-t)*x_2b

    w_1 = r*w_1a+(r-t)*w_1b
    w_2 = r*w_2a+(r-t)*w_2b

    if x_1 > w_1:
        return False # w_1 is not feasible
    elif x_2 > w_2:
        return False # w_2 is not feasible
    else:
        return True # w_1 and w_2 are feasible
```

In [ ]:
```
r = 2
found = False

while True:

    print(f"r = {r}")

    for t in range(1,r):
    #range function includes first value (1) but excludes last value (r).
    #this satisfies the requirement 0 < t < r
```

```
        g_a = g_allocation(x_a1,x_a2,w_a1,w_a2,r,t)

        is_Feasible = feasibility(g_a[0],g_a[1],x_b1,x_b2,r,t,w_a1,w_a2,w_b1,w_b2)

        # print(is_Feasible)
        Ut_g_a = g_a[0]*g_a[1]

        # print(f" Ut_g_a = {Ut_g_a}")

        Ut_w_a = w_a1*w_a2

        # print(f" Ut_w_a = {Ut_w_a}")

        Ut_x_a = x_a1*x_a2

        if is_Feasible == True:
            if Ut_g_a > Ut_w_a:
               if Ut_g_a > Ut_x_a:
                   print(f"a coalition with {r} A agents and {r-t} B agents object the given allocation.")
                   found = True
               else:
                   continue
            else:
                continue
        else:
            continue

    if found == True:
        break

    r = r + 1
```

```
r = 2
a coalition with 2 A agents and 1 B agents object the given allocation.
```

para este resultado esto fue una perdida de tiempo :)