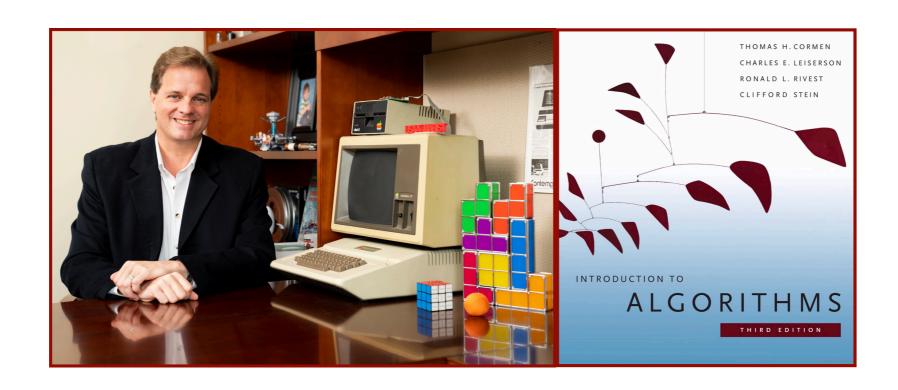
# Final Project

The Stable Marriage / Stable Matching / Matching Markets Problem



Alan G. Labouseur, Ph.D. Alan.Labouseur@Marist.edu

# Background, from Dr. Reid

The Stable Marriage / Stable Matching / Matching Markets Problem

#### The Stable Marriage Problem

Elizabeth Reid

Marist College Department of Mathematics Seminar Series

November 1, 2019

Elizabeth Reid

#### The Stable Marriage / Stable Matching / Matching Markets Problem

# Classic Set-up: • *n* men and *n* women each man lists the women according to preference each woman lists the men according to preference arrange marriages so no unmatched man and woman prefer each other to their assigned partners

Elizabeth Reid

#### The Stable Marriage / Stable Matching / Matching Markets Problem

# tiny Example

Two men: Alex and Bob;

Two women: Carol and Dana

 $\bullet$  A:C>D

• B: C > D

• C: A > B

• D: B > A

Elizabeth Reid

#### The Stable Marriage / Stable Matching / Matching Markets Problem

#### tiny Example

Two men: Alex and Bob;

Two women: Carol and Dana

• A: C > D

 $\bullet$  B: C > D

• C: A > B

• D: B > A

Possible Matchings:

 $\{(A,D),(B,C)\}$ 

 $\{(A,C),(B,D)\}$ 

#### The Stable Marriage / Stable Matching / Matching Markets Problem

#### tiny Example

Two men: Alex and Bob;

Two women: Carol and Dana

 $\bullet$  A:C>D

 $\bullet$  B: C > D

• C: A > B

• D: B > A

Possible Matchings:

 $\{(A,C),(B,D)\}$ 

#### The Stable Marriage / Stable Matching / Matching Markets Problem

#### tiny Example

Two men: Alex and Bob;

Two women: Carol and Dana

 $\bullet$  A:C>D

• B: C > D

• C: A > B

• D: B > A

Possible Matchings:

# The Stable Marriage / Stable Matching / Matching Markets Problem

#### Gale-Shapley Algorithm:

- Input.
  - A set of *n* men, a set of *n* women, a ranked list of the *n* women for each man, and a ranked list of the *n* men for each woman.

	1	2	3	4
Alex	Н	F	Е	G
Bob	F	G	Н	Ε
Carl	E	F	Н	G
Dave	Н	F	G	Ε

	1	2	3	4
Erin	В	Α	D	С
Fran	D	C	В	Α
Grace	Α	D	В	C
Holly	C	D	Α	В

Elizabeth Reid

#### The Stable Marriage / Stable Matching / Matching Markets Problem

#### Gale-Shapley Algorithm:

- Input.
  - A set of *n* men, a set of *n* women, a ranked list of the *n* women for each man, and a ranked list of the *n* men for each woman.

- Output.
  - A stable matching that pairs the n men and n women.

#### The Stable Marriage / Stable Matching / Matching Markets Problem

#### Gale-Shapley Algorithm:

- Input.
  - A set of *n* men, a set of *n* women, a ranked list of the *n* women for each man, and a ranked list of the *n* men for each woman.

- Output.
  - A stable matching that pairs the n men and n women.

When Men Propose:  $\{(A, E), (B, G), (C, F), (D, H)\}$ 

Does it make a difference who proposes?

Elizabeth Reid

#### The Stable Marriage / Stable Matching / Matching Markets Problem

#### Gale-Shapley Algorithm:

- Input.
  - A set of *n* men, a set of *n* women, a ranked list of the *n* women for each man, and a ranked list of the *n* men for each woman.

- Output.
  - A stable matching that pairs the n men and n women.

When Men Propose:  $\{(A, E), (B, G), (C, F), (D, H)\}$ When Women Propose:  $\{(A, G), (B, E), (C, H), (D, F)\}$ 

Elizabeth Reid

# Variation: Hospitals and Residents

The Stable Marriage / Stable Matching / Matching Markets Problem

#### Applications: Hospitals and Residents

- each medical student ranks hospital residency programs
- each hospital ranks the candidates

Elizabeth Reid

# Variation: Hospitals and Residents — Algorithm

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
assign all hospitals to be totally unsubscribed;
while (some resident r is free) and (r has a nonempty list) do
begin
    h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
    if h is fully subscribed then
         begin
             r' := worst resident provisionally assigned to h:
              assign r' to be free
         end:
    provisionally assign r to h;
    if h is fully subscribed then
         begin
              s := worst resident provisionally assigned to h;
              for each successor s' of s on h's list do
                   remove s' and h from each other's lists
          end
end
```

Figure 1.16: Resident-oriented algorithm

From page 42 of *The Stable Marriage Problem* by Gusfield and Irving. MIT Press. 1989

# Variation: Hospitals and Residents — Algorithm

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end ;
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                 r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5
                                 r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2
                                 r9: h4 h1 h5
   r4: h3 h4 h1 h5
                                 r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

Let's do it!

# Variation: Hospitals and Residents — Algorithm

# The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                 r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5
                                 r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2
                                 r9: h4 h1 h5
   r4: h3 h4 h1 h5
                                 r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

Initialize residents and hospitals.

All residents are free.

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                   Processing resident r1:
    while (some resident r is free) and (r has a nonempty list) do
                                                                   Provisionally assigning r1 to hospital h3.
    begin
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                   Processing resident r2:
       if h is fully subscribed then
                                                                   Provisionally assigning r2 to hospital h1.
           begin
              r' := worst resident provisionally assigned to h:
                                                                   Processing resident r3:
              assign r' to be free
                                                                   Provisionally assigning r3 to hospital h4.
           end:
       provisionally assign r to h;
                                                                   Processing resident r4:
        if h is fully subscribed then
                                                                   Provisionally assigning r4 to hospital h3.
           begin
              s := worst resident provisionally assigned to h:
                                                                   Processing resident r5:
               for each successor s' of s on h's list do
                                                                   Provisionally assigning r5 to hospital h1.
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                           Match:
   r1: (h3) h1 h5 h4
                               r7: h2 h5 h1 h3
                                                                                           (r1, h3)
   r2: h1 h3 h4 h2 h5
                               r8: h1 h3 h2 h5 h4
                               r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                               r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                    Processing resident r1:
    while (some resident r is free) and (r has a nonempty list) do
                                                                    Provisionally assigning r1 to hospital h3.
    begin
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                    Processing resident r2:
       if h is fully subscribed then
                                                                    Provisionally assigning r2 to hospital h1.
           begin
              r' := worst resident provisionally assigned to h:
                                                                    Processing resident r3:
               assign r' to be free
                                                                    Provisionally assigning r3 to hospital h4.
           end:
       provisionally assign r to h;
                                                                    Processing resident r4:
        if h is fully subscribed then
                                                                    Provisionally assigning r4 to hospital h3.
           begin
               s := worst resident provisionally assigned to h:
                                                                    Processing resident r5:
               for each successor s' of s on h's list do
                                                                   Provisionally assigning r5 to hospital h1.
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                           Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                           (r1, h3)
   r2: (h1) h3 h4 h2 h5
                                r8: h1 h3 h2 h5 h4
                                                                                           (r2, h1)
   r3: h4 h5 h3 h1 h2
                                r9: h4 h1 h5
   r4: h3 h4 h1 h5
                                r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                    Processing resident r1:
    while (some resident r is free) and (r has a nonempty list) do
                                                                   Provisionally assigning r1 to hospital h3.
    begin
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                    Processing resident r2:
       if h is fully subscribed then
                                                                   Provisionally assigning r2 to hospital h1.
           begin
              r' := worst resident provisionally assigned to h:
                                                                   Processing resident r3:
              assign r' to be free
                                                                    Provisionally assigning r3 to hospital h4.
           end:
       provisionally assign r to h;
                                                                   Processing resident r4:
        if h is fully subscribed then
                                                                   Provisionally assigning r4 to hospital h3.
           begin
              s := worst resident provisionally assigned to h:
                                                                    Processing resident r5:
               for each successor s' of s on h's list do
                                                                   Provisionally assigning r5 to hospital h1.
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                           Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                           (r1, h3)
   r2: h1 h3 h4 h2 h5
                               r8: h1 h3 h2 h5 h4
                                                                                           (r2, h1)
   r3: (h4) h5 h3 h1 h2
                                r9: h4 h1 h5
                                                                                           (r3, h4)
   r4: h3 h4 h1 h5
                                r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                    Processing resident r1:
    while (some resident r is free) and (r has a nonempty list) do
                                                                   Provisionally assigning r1 to hospital h3.
    begin
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                    Processing resident r2:
       if h is fully subscribed then
                                                                   Provisionally assigning r2 to hospital h1.
           begin
              r' := worst resident provisionally assigned to h:
                                                                    Processing resident r3:
              assign r' to be free
                                                                    Provisionally assigning r3 to hospital h4.
           end:
       provisionally assign r to h;
                                                                   Processing resident r4:
        if h is fully subscribed then
                                                                   Provisionally assigning r4 to hospital h3.
           begin
              s := worst resident provisionally assigned to h:
                                                                   Processing resident r5:
               for each successor s' of s on h's list do
                                                                   Provisionally assigning r5 to hospital h1.
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                           Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                           (r1, h3)
   r2: h1 h3 h4 h2 h5
                               r8: h1 h3 h2 h5 h4
                                                                                           (r2, h1)
                                r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
                                                                                           (r3, h4)
   r4: (h3) h4 h1 h5
                                r10: h3 h1 h5 h2 h4
                                                                                           (r4, h3)
   r5: h1 h4 h2
                                r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                    Processing resident r1:
    while (some resident r is free) and (r has a nonempty list) do
                                                                    Provisionally assigning r1 to hospital h3.
    begin
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                    Processing resident r2:
       if h is fully subscribed then
                                                                    Provisionally assigning r2 to hospital h1.
           begin
              r' := worst resident provisionally assigned to h:
                                                                    Processing resident r3:
               assign r' to be free
                                                                    Provisionally assigning r3 to hospital h4.
           end:
       provisionally assign r to h;
                                                                    Processing resident r4:
        if h is fully subscribed then
                                                                    Provisionally assigning r4 to hospital h3.
           begin
               s := worst resident provisionally assigned to h:
                                                                    Processing resident r5:
               for each successor s' of s on h's list do
                                                                    Provisionally assigning r5 to hospital h1.
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                           Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                           (r1, h3)
   r2: h1 h3 h4 h2 h5
                                r8: h1 h3 h2 h5 h4
                                                                                           (r2, h1)
                                r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
                                                                                           (r3, h4)
   r4: h3 h4 h1 h5
                                r10: h3 h1 h5 h2 h4
                                                                                           (r4, h3)
   r5: (h1) h4 h2
                                r11: h5 h4 h1 h3 h2
                                                                                           (r5, h1)
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                             Wow... this is really easy! I bet it never...
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                 r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5
                                 r8: h1 h3 h2 h5 h4
   r3: h4 h5 h3 h1 h2
                                 r9: h4 h1 h5
   r4: h3 h4 h1 h5
                                 r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: (h4) h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h4.

Hospital h4 is full. Finding worst resident assigned to hospital h4.

Considering r8. (no), Considering r6. (yes)

Removing all residents after r6 from hospital h4's list:

Deleting (h4, r8): Removing hospital h4 from resident r8's list.

Removing r8 from hospital h4's list.
```

#### Match: (r1, h3) (r2, h1) (r3, h4) (r4, h3) (r5, h1) (r6, h4)

Oh.

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h :=
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                 r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5
                                 r8: h1 h3 h2 h5 h4
   r3: (h4) h5 h3 h1 h2
                                 r9: h4 h1 h5
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: (h4) h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 x11 r9
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h4.

Hospital h4 is full. Finding worst resident assigned to hospital h4.

Considering r8. (no), Considering r6. (yes)

Removing all residents after r6 from hospital h4's list:

Deleting (h4, r8): Removing hospital h4 from resident r8's list.

Removing r8 from hospital h4's list.
```

```
Match:

(r1, h3)

(r2, h1)

(r3, h4)

(r4, h3)

(r5, h1)

(r6, h4)
```

Never mind. I see it now.

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                  r7: h2 h5 h1 h3
                                  r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
                                  r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h4.
Hospital h4 is full. Finding worst resident assigned to hospital h4.
Considering r8. (no), Considering r6. (yes)
Removing all residents after r6 from hospital h4's list:
Deleting (h4, r8): Removing hospital h4 from resident r8's list.
Removing r8 from hospital h4's list.
```

# Match: (r1, h3) (r2, h1) (r3, h4) (r4, h3) (r5, h1) (r6, h4)

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
                assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
                                  r7: (h2) h5 h1 h3
   r1: h3 h1 h5 h4
   r2: h1 h3 h4 h2 h5
                                  r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r7:
Provisionally assigning r7 to hospital h2.
```

Processing resident r8:
Provisionally assigning r8 to hospital h1.

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r7, h2)
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
                assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                  r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5
                                  r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

Processing resident r7:
Provisionally assigning r7 to hospital h2.

Processing resident r8:
Provisionally assigning r8 to hospital h1.

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r7, h2)
(r8, h1)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                  Processing resident r9: Hospital h4 is full.
    while (some resident r is free) and (r has a nonempty list) do
                                                                  Finding worst resident assigned to hospital h4.
    begin
                                                                     Considering r6. (yes)
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                     Bumping r6 from hospital h4.
       if h is fully subscribed then
                                                                  Provisionally assigning r9 to hospital h4.
           begin
                                                                  Hospital h4 is (still) full. Finding worst resident
              r' := worst resident provisionally assigned to h:
                                                                  assigned to hospital h4.
              assign r' to be free
                                                                     Considering r6. (no) Considering r3. (yes)
           end:
                                                                      Removing all residents after r3 from
       provisionally assign r to h;
                                                                     hospital h4's list:
       if h is fully subscribed then
                                                                         Deleting (h4, r6): Removing hospital h4
           begin
                                                                         from resident r6's list.
              s := worst resident provisionally assigned to h:
                                                                         Removing r6 from hospital h4's list.
              for each successor s' of s on h's list do
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                         Match:
   r1: h3 h1 h5 h4
                               r7: h2 h5 h1 h3
                                                                                         (r1, h3)
                               r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
                                                                                         (r2, h1)
                               r9: h4 h1 h5
   r3: (h4) h5 h3 h1 h2
                                                                                         (r3, h4)
   r4: h3 h4 h1 h5
                               r10: h3 h1 h5 h2 h4
                                                                                         (r4, h3)
   r5: h1 h4 h2
                               r11: h5 h4 h1 h3 h2
                                                                                         (r5, h1)
   r6: (h4) h3 h2 h1 h5
                                                                                                      (make room in h4 for r9)
                                                                                         (r6, h4)
Hospital Preferences:
                                                                                         (r7, h2)
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                         (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 x11
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                  Processing resident r9: h4 is full
    while (some resident r is free) and (r has a nonempty list) do
                                                                  Finding worst resident assigned to hospital h4.
    begin
                                                                     Considering r6. (yes)
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                     Bumping r6 from hospital h4.
       if h is fully subscribed then
                                                                  Provisionally assigning r9 to hospital h4.
           begin
                                                                  Hospital h4 is (still) full. Finding worst resident
              r' := worst resident provisionally assigned to h:
                                                                  assigned to hospital h4.
              assign r' to be free
                                                                     Considering r6. (no) Considering r3. (yes)
           end:
                                                                     Removing all residents after r3 from
       provisionally assign r to h;
                                                                     hospital h4's list:
       if h is fully subscribed then
                                                                         Deleting (h4, r6): Removing hospital h4
           begin
                                                                         from resident r6's list.
              s := worst resident provisionally assigned to h:
                                                                         Removing r6 from hospital h4's list.
              for each successor s' of s on h's list do
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                         Match:
                               r7: h2 h5 h1 h3
   r1: h3 h1 h5 h4
                                                                                         (r1, h3)
   r2: h1 h3 h4 h2 h5
                                                                                         (r2, h1)
                               r9: (h4) h1 h5
   r3: h4 h5 h3 h1 h2
                                                                                         (r3, h4)
   r4: h3 h4 h1 h5
                               r10: h3 h1 h5 h2 h4
                                                                                         (r4, h3)
                               r11: h5 h4 h1 h3 h2
   r5: h1 h4 h2
                                                                                         (r5, h1)
   r6: h4 h3 h2 h1 h5
                                                                                         (r6, h4)
Hospital Preferences:
                                                                                         (r7, h2)
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                         (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
                                                                                         (r9, h4)
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
                                  r7: h2 h5 h1 h3
   r1: h3 h1 h5 h4
   r2: h1 h3 h4 h2 h5
   r3: (h4) h5 h3 h1 h2
                                  r9: (h4) h1 h5
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
   r5: h1 h4 h2
                                  r11: h5 h4 h1 h3 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 t3 r11 r9
```

```
Processing resident r9: h4 is full
Finding worst resident assigned to hospital h4.

Considering r6. (yes)

Bumping r6 from hospital h4.

Provisionally assigning r9 to hospital h4.

Hospital h4 is (still) full. Finding worst resident assigned to hospital h4.

Considering r6. (no) Considering r3. (yes)

Removing all residents after r3 from hospital h4's list:

Deleting (h4, r6): Removing hospital h4 from resident r6's list.

Removing r6 from hospital h4's list.
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4)
(r6 is still unassigned)
(r7, h2)
(r8, h1)
(r9, h4)
```

#### The Stable Marriage / Stable Matching / Matching Markets Problem

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h := 
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
                                  r7: h2 h5 h1 h3
   r1: h3 h1 h5 h4
   r2: h1 h3 h4 h2 h5
                                  r9: (h4) h1 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
                                  r11: h5 h4 h1 h3 h2
   r5: h1 h4 h2
   r6: h4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 x10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r9: h4 is full
Finding worst resident assigned to hospital h4.

Considering r6. (yes)

Bumping r6 from hospital h4.

Provisionally assigning r9 to hospital h4.

Hospital h4 is (still) full. Finding worst resident assigned to hospital h4.

Considering r6. (no) Considering r3. (yes)

Removing all residents after r3 from hospital h4's list:

Deleting (h4, r6): Removing hospital h4 from resident r6's list.
```

Removing r6 from hospital h4's list.

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h4) (r6 is still unassigned)
(r7, h2)
(r8, h1)
(r9, h4)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                  r7: h2 h5 h1 h3
                                  r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
                                  r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
                                  r11: h5 h4 h1 h3 h2
   r5: h1 h4 h2
          (h3) h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 x10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
Processing resident r6:
Provisionally assigning r6 to hospital h3.

Hospital h3 is full. Finding worst resident assigned to hospital h3.

Considering r10. (no), Considering r1. (yes)
Removing all residents after r1 from hospital h3's list:

Deleting (h3, r10): Removing hospital h3 from resident r10's list.

Removing r10 from hospital h3's list.
```

```
Match:
  (r1, h3)
  (r2, h1)
  (r3, h4)
  (r4, h3)
  (r5, h1)
  (r6, h3)  (r6 now reassigned)
  (r7, h2)
  (r8, h1)
  (r9, h4)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                   Processing resident r6:
    while (some resident r is free) and (r has a nonempty list) do
                                                                   Provisionally assigning r6 to hospital h3.
    begin
                                                                   Hospital h3 is full. Finding worst resident assigned
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                   to hospital h3.
       if h is fully subscribed then
                                                                      Considering r10. (no), Considering r1. (yes)
           begin
                                                                      Removing all residents after r1 from
              r' := worst resident provisionally assigned to h:
                                                                      hospital h3's list:
              assign r' to be free
                                                                          Deleting (h3, r10): Removing hospital h3
           end:
                                                                          from resident r10's list.
       provisionally assign r to h;
                                                                          Removing r10 from hospital h3's list.
       if h is fully subscribed then
           begin
              s := worst resident provisionally assigned to h:
               for each successor s' of s on h's list do
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                          Match:
   r1: (h3) h1 h5 h4
                               r7: h2 h5 h1 h3
                                                                                          (r1, h3)
                               r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
                                                                                          (r2, h1)
                               r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
                                                                                          (r3, h4)
   r4: (h3) h4 h1 h5
                               r10: h3 h1 h5 h2 h4
                                                                                          (r4, h3)
   r5: h1 h4 h2
                               r11: h5 h4 h1 h3 h2
                                                                                           (r5, h1)
   r6: h4(h3) h2 h1 h5
                                                                                          (r6, h3)
                                                                                          (r7, h2)
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                          (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 x11
                                                                                          (r9, h4)
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1 r10
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r6
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                   Processing resident r6:
    while (some resident r is free) and (r has a nonempty list) do
                                                                   Provisionally assigning r6 to hospital h3.
    begin
                                                                   Hospital h3 is full. Finding worst resident assigned
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                   to hospital h3.
       if h is fully subscribed then
                                                                      Considering r10. (no), Considering r1. (yes)
           begin
                                                                      Removing all residents after r1 from
              r' := worst resident provisionally assigned to h:
                                                                      hospital h3's list:
              assign r' to be free
                                                                          Deleting (h3, r10): Removing hospital h3
           end:
                                                                          from resident r10's list.
       provisionally assign r to h;
                                                                          Removing r10 from hospital h3's list.
       if h is fully subscribed then
           begin
              s := worst resident provisionally assigned to h:
               for each successor s' of s on h's list do
                  remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                          Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                          (r1, h3)
                               r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
                                                                                          (r2, h1)
                                r9: h4 h1 h5
   r3: h4 h5 h3 h1 h2
                                                                                          (r3, h4)
                               r10: h3 h1 h5 h2 h4
   r4: h3 h4 h1 h5
                                                                                          (r4, h3)
                                r11: h5 h4 h1 h3 h2
   r5: h1 h4 h2
                                                                                          (r5, h1)
         (h3) h2 h1 h5
                                                                                          (r6, h3)
Hospital Preferences:
                                                                                          (r7, h2)
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                          (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
                                                                                          (r9, h4)
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 x8 x
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
                assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
                s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                    remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                  r7: h2 h5 h1 h3
                                  r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3(h1) h5 h2 h4
                                  r11: h5 h4 h1 h3 h2
   r5: h1 h4 h2
          4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
Processing resident r10:

Provisionally assigning r10 to hospital h1.

Hospital h1 is full. Finding worst resident assigned to hospital h1.

Considering r2. (yes)

Removing all residents after r2

from hospital h1's list:
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                     Processing resident r10:
    while (some resident r is free) and (r has a nonempty list) do
                                                                     Provisionally assigning r10 to hospital h1.
    begin
                                                                     Hospital h1 is full. Finding worst resident assigned
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                     to hospital h1.
       if h is fully subscribed then
                                                                        Considering r2. (yes)
           begin
                                                                        Removing all residents after r2
              r' := worst resident provisionally assigned to h:
                                                                        from hospital h1's list:
               assign r' to be free
           end:
        provisionally assign r to h;
        if h is fully subscribed then
           begin
               s := worst resident provisionally assigned to h :=
               for each successor s' of s on h's list do
                   remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                             Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                             (r1, h3)
   r2: (h1) h3 h4 h2 h5
                                                                                             (r2, h1)
   r3: h4 h5 h3 h1 h2
                                                                                             (r3, h4)
   r4: h3 h4 h1 h5
                                      h3(h1) h5 h2 h4
                                                                                             (r4, h3)
   r5: (h1) h4 h2
                                                                                             (r5, h1)
   r6: h4 h3 h2 h1 h5
                                                                                             (r6, h3)
Hospital Preferences:
                                                                                             (r7, h2)
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6
                                                                                             (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
                                                                                             (r9, h4)
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
                                                                                             (r10, h1)
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                     Processing resident r10:
    while (some resident r is free) and (r has a nonempty list) do
                                                                    Provisionally assigning r10 to hospital h1.
    begin
                                                                     Hospital h1 is full. Finding worst resident assigned
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
                                                                     to hospital h1.
       if h is fully subscribed then
                                                                        Considering r2. (yes)
           begin
                                                                        Removing all residents after r2
              r' := worst resident provisionally assigned to h:
                                                                        from hospital h1's list:
               assign r' to be free
           end:
       provisionally assign r to h;
       if h is fully subscribed then
           begin
               s := worst resident provisionally assigned to h:
               for each successor s' of s on h's list do
                   remove s' and h from each other's lists
           end
    end
Resident Preferences:
                                                                                            Match:
   r1: h3 h1 h5 h4
                                r7: h2 h5 h1 h3
                                                                                            (r1, h3)
   r2: (h1) h3 h4 h2 h5
                                                                                            (r2, h1)
   r3: h4 h5 h3 h1 h2
                                                                                            (r3, h4)
   r4: h3 h4 h1 h5
                                      h3(h1) h5 h2 h4
                                                                                            (r4, h3)
   r5: (h1) h4 h2
                                                                                            (r5, h1)
   r6: h4 h3 h2 h1 h5
                                                                                            (r6, h3)
Hospital Preferences:
                                                                                            (r7, h2)
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                            (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
                                                                                            (r9, h4)
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
                                                                                            (r10, h1)
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
                assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
                s := worst resident provisionally assigned to h:
                for each successor s' of s on h's list do
                    remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                  r7: h2 h5 h1 h3
                                  r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
                                  r11: (h5) h4 h1 h3 h2
   r5: h1 h4 h2
          4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3
```

```
Processing resident r11:

Provisionally assigning r11 to hospital h5.

Hospital h5 is full. Finding worst resident assigned to hospital h5.

Considering r9. (no), Considering r11. (yes)

Removing all residents after r11

from hospital h5's list:

Deleting (h5, r9): Removing hospital h5 from resident r9's list.

Removing r9 from hospital h5's list.
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
                assign r' to be free
            end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
                s := worst resident provisionally assigned to h :=
                for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                  r7: h2 h5 h1 h3
                                  r8: h1 h3 h2 h5
   r2: h1 h3 h4 h2 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
                                  r10: h3 h1 h5 h2 h4
                                  r11: (h5) h4 h1 h3 h2
   r5: h1 h4 h2
          4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3 r8 r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11 r9
```

```
Processing resident r11:
Provisionally assigning r11 to hospital h5.

Hospital h5 is full. Finding worst resident assigned to hospital h5.

Considering r9. (no), Considering r11. (yes)

Removing all residents after r11

from hospital h5's list:

Deleting (h5, r9): Removing hospital h5 from resident r9's list.

Removing r9 from hospital h5's list.
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
    while (some resident r is free) and (r has a nonempty list) do
    begin
        h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
        if h is fully subscribed then
            begin
               r' := worst resident provisionally assigned to h:
                assign r' to be free
            end ;
        provisionally assign r to h;
        if h is fully subscribed then
            begin
                s := worst resident provisionally assigned to h := 
                for each successor s' of s on h's list do
                    remove s' and h from each other's lists
            end
     end
Resident Preferences:
   r1: h3 h1 h5 h4
                                   r7: h2 h5 h1 h3
   r2: h1 h3 h4 h2 h5
   r3: h4 h5 h3 h1 h2
   r4: h3 h4 h1 h5
   r5: h1 h4 h2
                                             h4 h1 h3 h2
          4 h3 h2 h1 h5
Hospital Preferences:
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3/r8/r8
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 r11
```

```
Processing resident r11:

Provisionally assigning r11 to hospital h5.

Hospital h5 is full. Finding worst resident assigned to hospital h5.

Considering r9. (no), Considering r11. (yes)

Removing all residents after r11

from hospital h5's list:

Deleting (h5, r9): Removing hospital h5 from resident r9's list.

Removing r9 from hospital h5's list.
```

```
Match:
(r1, h3)
(r2, h1)
(r3, h4)
(r4, h3)
(r5, h1)
(r6, h3)
(r7, h2)
(r8, h1)
(r9, h4)
(r10, h1)
(r11, h5)
```

```
assign all residents to be free;
    assign all hospitals to be totally unsubscribed:
                                                                      No residents are free.
    while (some resident r is free) and (r has a nonempty list) do
    begin
       h := first hospital on r's list ; \{r \text{ "proposes" to } h\}
       if h is fully subscribed then
           begin
               r' := worst resident provisionally assigned to h:
               assign r' to be free
           end:
        provisionally assign r to h;
        if h is fully subscribed then
            begin
               s := worst resident provisionally assigned to h:
               for each successor s' of s on h's list do
                   remove s' and h from each other's lists
            end
     end
Resident Preferences:
                                                                                               Final Stable Match:
   r1: h3 h1 h5 h4
                                                                                               (r1, h3)
   r2: h1 h3 h4 h2 h5
                                                                                               (r2, h1)
   r3: h4 h5 h3 h1 h2
                                                                                               (r3, h4)
   r4: h3 h4 h1 h5
                                                                                               (r4, h3)
   r5: h1 h4 h2
                                                                                               (r5, h1)
          1 h3 h2 h1 h5
                                                                                               (r6, h3)
Hospital Preferences:
                                                                                               (r7, h2)
   h1 (capacity 4): r3 r7 r9 r11 r5 r4 r10 r8 r6 r1 r2
                                                                                               (r8, h1)
   h2 (capacity 3): r5 r7 r10 r6 r8 r2 r3 r11
                                                                                               (r9, h4)
   h3 (capacity 3): r11 r6 r8 r3 r2 r4 r7 r1
                                                                                               (r10, h1)
   h4 (capacity 2): r10 r1 r2 r11 r4 r9 r5 r3
                                                                                               (r11, h5)
   h5 (capacity 1): r2 r4 r10 r7 r6 r1 r8 r3 Y1
```

# Final Project

The Stable Marriage / Stable Matching / Matching Markets Problem

Part one:

(70 points)

Implement the Hospital and Residents variation as shown here.

- Test your code on this data, Make sure your results match mine.
- Then test your code on your own data and make a case that it's correct.

Part two:

(20 points)

Implement a variation on the Hospital and Residents variation where to hospitals do not rank residents but residents still rank the hospitals.

- Define what "stability" means in this context.
- Then test your code on your own data and make a case that it's correct.

See the final project assignment on our class web site for additional details (like the remaining 10 points).