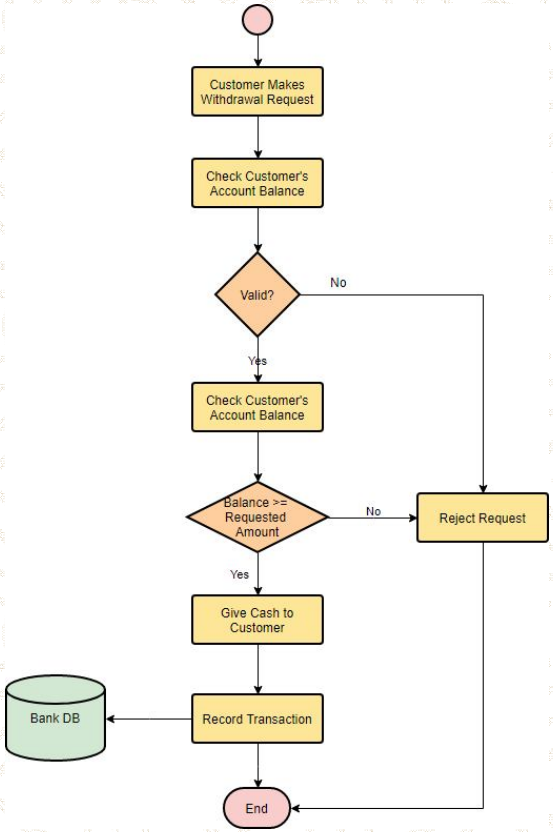
# ALGORITHMS

Dr. H. İrem TÜRKMEN

# Life Cycle Of a Software

- Problem
- Algorithm
  - Flowchart
  - Pseudo Code
- Analysis
- Implementation
- Test
- Production




```
set total to zero

get list of numbers

loop through each number in the list
   add each number to total
end loop

if number more than zero
   print "it's positive" message
else
   print "it's zero or less" message
end if
```

# Defination

- The central concept underlying all computation is that of the algorithm
- An algorithm is a ***step-by-step sequence of instructions for carrying out some task***
- Programming can be viewed as the process of designing and implementing algorithms that a computer can carry out.
- A programmer's job is to:

  Create an algorithm for accomplishing a given objective, then translate the individual steps of the algorithm into a programming language that the computer can understand

# Algorithms

- Algorithms are well-defined sequence of unambiguous instructions
- Must terminate (to produce a result)
- Algorithm description relies on a well-defined “instruction language”

- **Example: Manual Addition**

  *Describe the method!*

  123456

+ 789001

  912457

# Algorithms

- The use of algorithms is not limited to the domain of computing
  - e.g., recipes for baking cookies
  - e.g., directions to your house
- There are many unfamiliar tasks in life that we could not complete without the aid of instructions
- In order for an algorithm to be effective, it must be stated in a manner that its intended executor can understand
- A **recipe written for a master chef will look different than a recipe written for a college student**
- As you have already experienced, computers are more demanding with regard to algorithm specifics than any human could be

# Algorithms

- Obtain a basket of unshelled peas and an empty bowl.
- As long as there are unshelled peas in the basket continue to execute the following steps:

  a. Take a pea from the basket.

  b. Break open the pea pod.

  c. Dump the peas from the pod into the bowl.

  d. Discard the pod.

# Designing & Analyzing Algorithms

- Steps to solving problems
  1. understand the problem
  2. devise a plan
  3. carry out your plan
  4. examine the solution
- **EXAMPLE: finding the oldest** person in a room full of people

- **Understanding the problem**

  Initial condition: room full of people

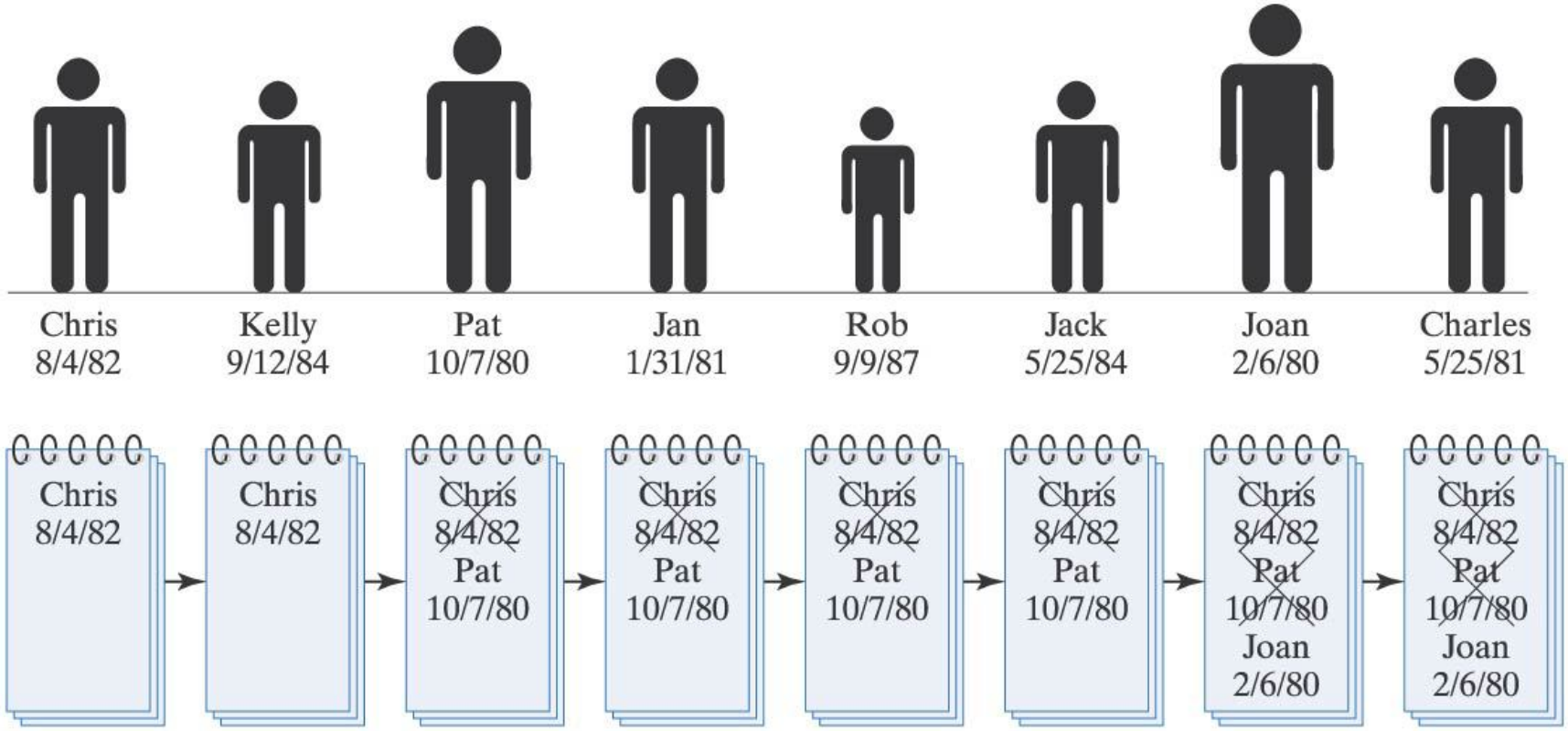
  Goal: identify the oldest person

  Assumptions:

  - a person will give their real birthday
  - two people are born on the same day, they are the same age
  - if there is more than one oldest person, finding any one of them is okay

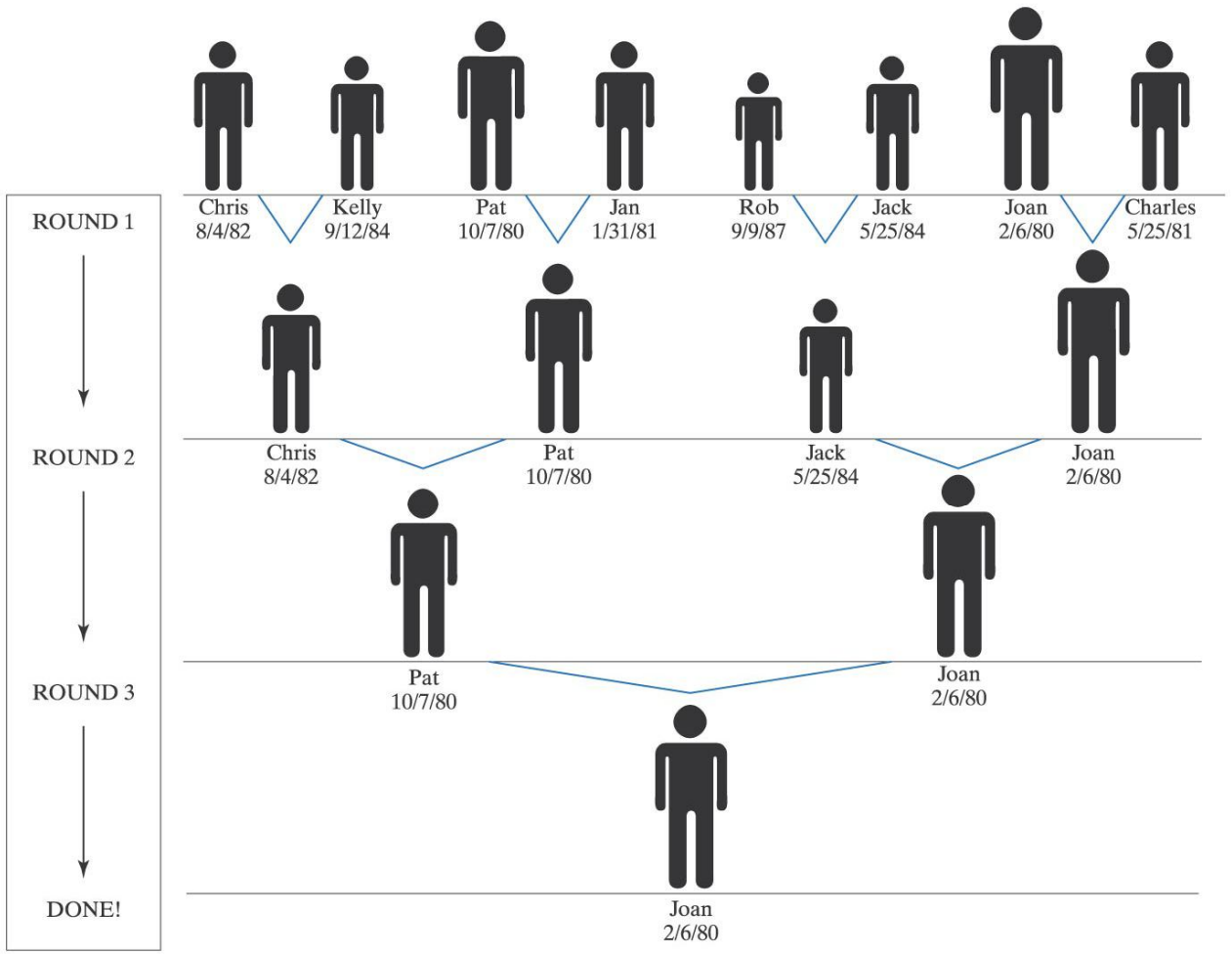
# ALGORITHM #1

1. line up all the people along one wall
2. ask the first person to state their name and birthday, then write this information down on a piece of paper
3. for each successive person in line:
    i. ask the person for their name and birthday
    ii. if the stated birthday is earlier than the birthday on the paper, cross out old information and write down the name and birthday of this person
4. **When you reach the end of the line, the name and birthday of the oldest person will be written on the paper**



# ALGORITHM #2

1. line up all the people along one wall
2. as long as there is more than one person in the line, repeatedly
   i. have the people pair up (1st with 2nd, 3rd with 4th, etc) – if there are an odd number of people, the last person will be without a partner
   ii. ask each pair of people to compare their birthdays
   iii. request that the younger of the two leave the line
3. **When there is only one person left in line, that person is the oldest**



# ALGORITHM ANALYSIS

- Determining which algorithm is "better" is not always clear cut
- It depends upon what features are most important to you
    - if you want to be sure it works, choose the /clearer algorithm
    - if you care about the time or effort required, need to analyze performance

# ANALYSIS of ALGORITHM#1

- Algorithm 1 involves asking each person's birthday and then comparing it to the birthday written on the page
- The amount of time to find the oldest person is ***proportional to the number of people***
- If you double the amount of people, the time needed to find the oldest person will also double

# ANALYSIS of ALGORITHM#2

- Algorithm 2 allows you to perform multiple comparisons simultaneously
- The time needed to find the oldest person is ***proportional to the number of rounds it takes to shrink the line down to one person***
- Which turns out to be the logarithm (base 2) of the number of people
- If you double the amount of people, the time needed to find the oldest person increases by a factor of one more comparison

# ALGORITHM ANALYSIS

- **Assume it takes 5 seconds to compare birthdays**

- For algorithm 1:

100 people 5*100 = 500 seconds

200 people 5*200 = 1000 seconds

400 people 5*400 = 2000 seconds

. . .

1,000,000 people 5*1,000,000 = 5,000,000 seconds

- **When the problem size is large, performance differences can be dramatic**

- For algorithm 2:

100 people 5*$\log_2$100 = 35 seconds

200 people 5* $\log_2$ 200 = 40 seconds

400 people 5* $\log_2$ 400 = 45 seconds

. . .

1,000,000 people 5* $\log_2$ 1,000,000 = 100 seconds