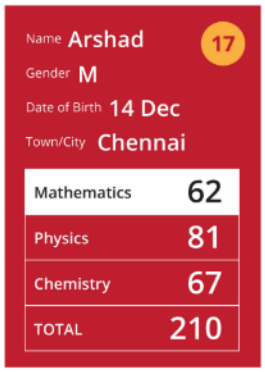
CT Week 3 : Multiple iterations (non-nested), Three prizes problem, Procedures, Parameters, Side effects, OR operator

Lecture 1 : Presentation of datasets in the form of a table

Tables

Extracting data from cards

- Each card is a unit of information
- Different attributes or fields
  - Card ID, Name, Gender, . . . , Total
- Organize as a table
- All the grade cards in a single table



Id	Name	Gen	DoB	City	Math	Phys	Chem	Total
10	Kavya	F	12 Jan	Chennai	64	72	68	204
24	Siddharth	M	26 Dec	Madurai	44	72	58	174
17	Arshad	M	14 Dec	Chennai	62	81	67	210

Word in the paragraph

- Can do the same with words

Id	Word	Type	Length
24	Saturday	Noun	8
12	considered	Verb	10

Shopping bills

- What are the attributes?
  - Bill ID, Shop Name, Customer Name
  - Item, Category, Qty, Price, Cost, Total
    - Variable number of rows per bill
- Variable number of columns?
  - No longer a neat table!
- Variable number of rows per card?
  - Tag rows for each card

Shopping bill as a table

Id	Shop	Customer	Item	Category	Qty	Price	Cost	Total
5	Big Bazaar	Akshaya	Trousers	Women/Apparel	2	870	1740	4174
5	Big Bazaar	Akshaya	Shirts	Women/Apparel	1	1350	1350	4174
5	Big Bazaar	Akshaya	Detergent	Household	0.5	270	135	4174
5	Big Bazaar	Akshaya	Tee Shirts	Women/Apparel	4	220	880	4174
5	Big Bazaar	Akshaya	Instant Noodles	Canned/Food	3	23	69	4174

- Id, Shop, Customer and Total are duplicated in each row of the table

Big Bazaar — Akshaya — 5

Item	Category	Qty	Price	Cost
Trousers	Women/Apparel	2	870	1740
Shirts	Women/Apparel	1	1350	1350
Detergent	Household	0.5	270	135
Tee shirts	Women/Apparel	4	220	880
Instant Noodles	Canned/Food	3	23	69
				4174

Id	Shop	Customer	Item	Category	Qty	Price	Cost	Total
5	Big Bazaar	Akshaya	Trousers	Women/Apparel	2	870	1740	4174
5	Big Bazaar	Akshaya	Shirts	Women/Apparel	1	1350	1350	4174
5	Big Bazaar	Akshaya	Detergent	Household	0.5	270	135	4174
5	Big Bazaar	Akshaya	Tee Shirts	Women/Apparel	4	220	880	4174
5	Big Bazaar	Akshaya	Instant Noodles	Canned/Food	3	23	69	4174

- Id, Shop, Customer and Total are duplicated in each row of the table
- Problem is aggravated as number of entries in the bill increase

SV Stores — Srivatsan — 1

Item	Category	Qty	Price	Cost
Carrots	Vegetables/Food	1.5	50	75
Soap	Toiletries	4	32	128
Tomatoes	Vegetables/Food	2	40	80
Bananas	Vegetables/Food	8	8	64
Socks	Footwear/Apparel	3	56	168
Curd	Dairy/Food	0.5	32	16
Milk	Dairy/Food	1.5	24	36
				567

Multiple tables

Id	Shop	Customer	Total
1	SV Stores	Srivatsan	567

Id	Item	Category	Qty	Price	Cost
1	Carrots	Vegetables/Food	1.5	50	75
1	Soap	Toiletries	4	32	128
1	Tomatoes	Vegetables/Food	2	40	80
		⋮			
1	Milk	Dairy/Food	1.5	24	36

- One table has columns that are fixed for the bill
- Second table has variable entries as multiple rows
- Bill ID links the tables

SV Stores — Srivatsan (1)

Item	Category	Qty	Price	Cost
Carrots	Vegetables/Food	1.5	50	75
Soap	Toiletries	4	32	128
Tomatoes	Vegetables/Food	2	40	80
Bananas	Vegetables/Food	8	8	64
Socks	Footwear/Apparel	3	56	168
Curd	Dairy/Food	0.5	32	16
Milk	Dairy/Food	1.5	24	36
				567

Id	Shop	Customer	Total
5	Big Bazaar	Akshaya	4174

Id	Item	Category	Qty	Price	Cost
5	Trousers	Women/Apparel	2	870	1740
5	Shirts	Women/Apparel	1	1350	1350
5	Detergent	Household	0.5	270	135
5	Tee Shirts	Women/Apparel	4	220	880
5	Instant Noodles	Canned/Food	3	23	69

- One table has columns that are fixed for the bill
- Second table has variable entries as multiple rows
- Bill ID links the tables

Big Bazaar — Akshaya (5)

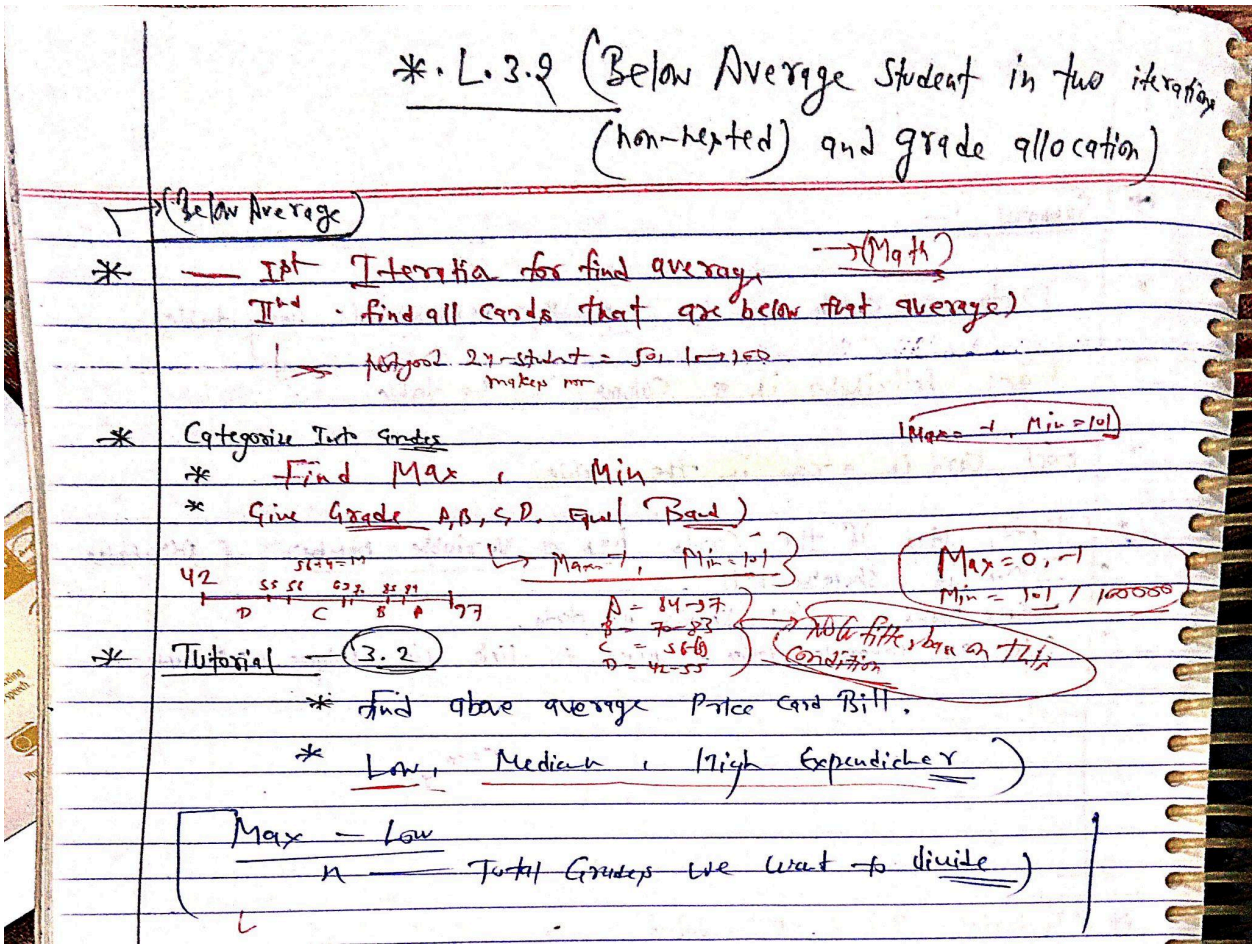
Item	Category	Qty	Price	Cost
Trousers	Women/Apparel	2	870	1740
Shirts	Women/Apparel	1	1350	1350
Detergent	Household	0.5	270	135
Tee shirts	Women/Apparel	4	220	880
Instant Noodles	Canned/Food	3	23	69
				4174

Summary

- Data on cards can be naturally represented using tables
- Each attribute is a column in the table
- Each card is a row in the table
- Difficulty if the cards has a variable number of attributes
  - Items in shopping bill
  - Multiple rows — duplication of data
  - Split as separate tables — need to link via unique attribute

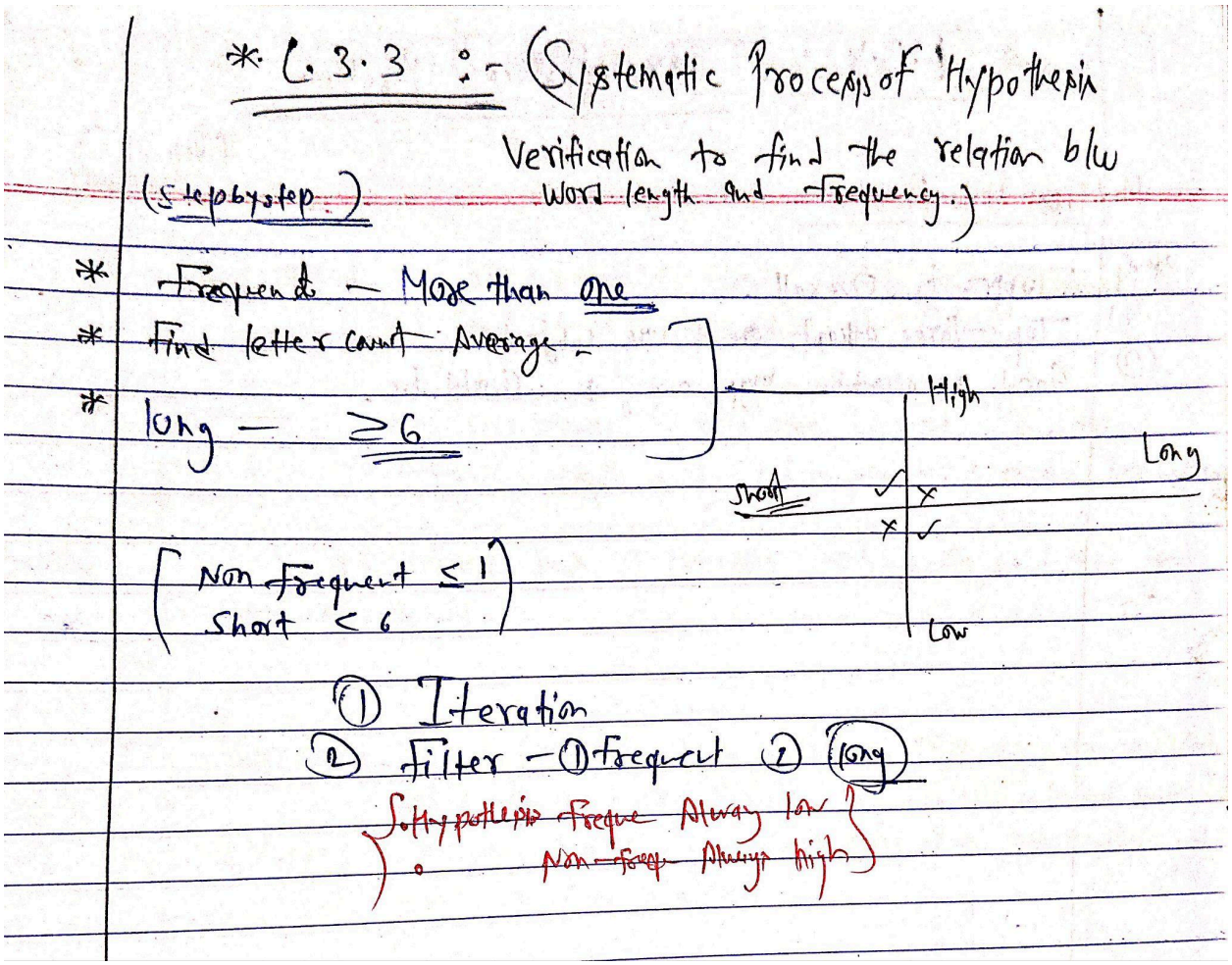
QN - 5,6

Lecture 2 : Below average students in two iterations (non-nested) and grade allocation



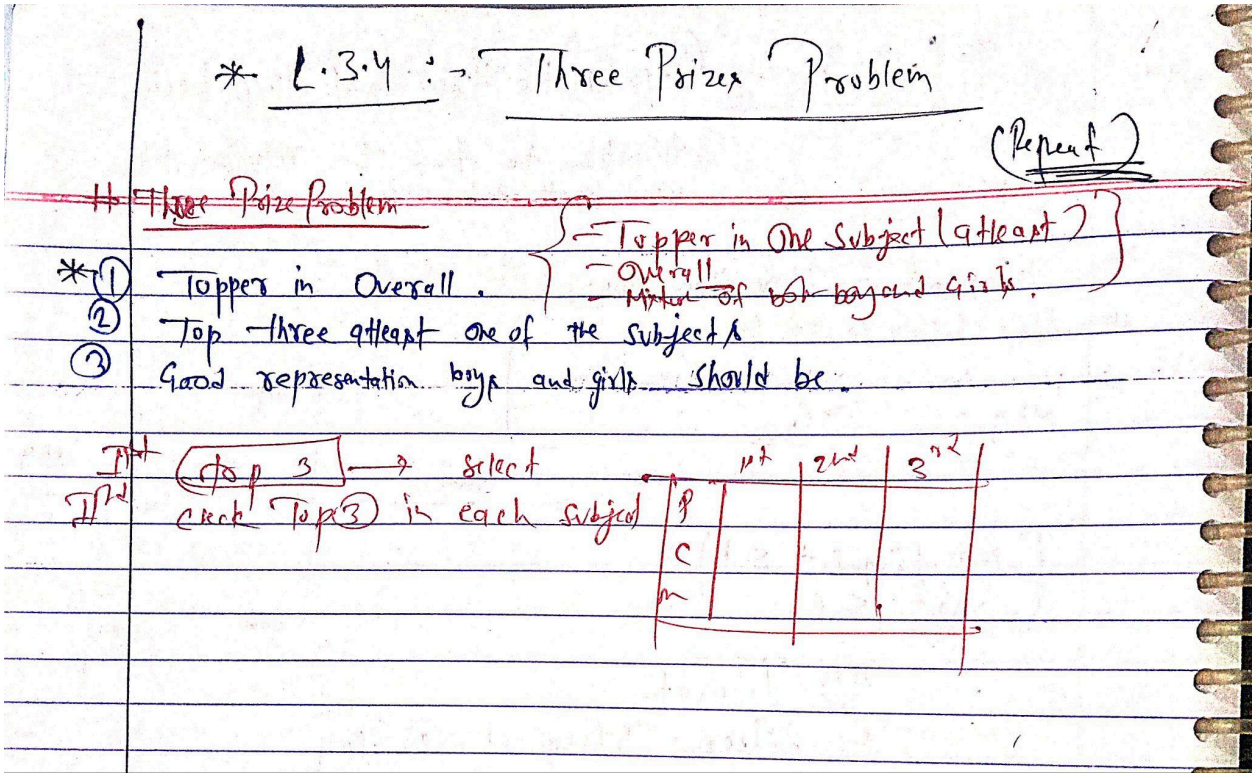
QN : 1

Lecture 3 : Systematic process of hypothesis verification to find the relation between word length and frequency



QN : 3,4,7

Lecture 4 : Three prizes problem



QN : 2,3,9

Lecture 5 : Introduction to procedures and parameters

A **procedure** is a **set of steps or actions** that are followed in a particular order to achieve a result.

Example:

- A doctor follows a **medical procedure** during surgery.
- A company has a **procedure** for hiring new employees.

A procedure is a *systematic series of actions* directed toward a specific goal.

In Programming

In programming, a **procedure** is a **block of code** designed to perform a specific task.
It's similar to a **function**, but may or may not return a value — depending on the language.

* Procedures :- We are talking about Procapp and Procedure
  → same set of Instruction which it follow.

* Parameter :- When we are passing the field name to the Procedures to make it adapt that we in particular called parameter.

(Procedures has a parameter which is the field that you want to look at.)

*. Procedures :- A Procedure is a block of organized, reusable code that is used to perform a single, related action. Procedure provide better modularity for your application and a high degree of code reusing.

- Identify which Part can be Modularised.
- Define Procedure.
- Define body.
- Call Procedure.

Procedure → Define – Parameter – (Table)
Call – Argument → Value पास करें।

1. A Procedure may not return a Value.
2. Procedure call is a separate statement.
3. Use a procedure when the same computation is used for different situations. (M,P), (M,C), (F,P)
4. Parameters fix the context. (gen, field)
5. Use variables to save Values returned by procedures
6. keep track of the outcomes of multiple procedure calls.
7. Procedure help to modularize pseudocode.
8. Avoid describing the same process repeatedly.
9. If we improve the code in a Procedure, benefit automatically applies to all procedure calls.

QN : 2,4

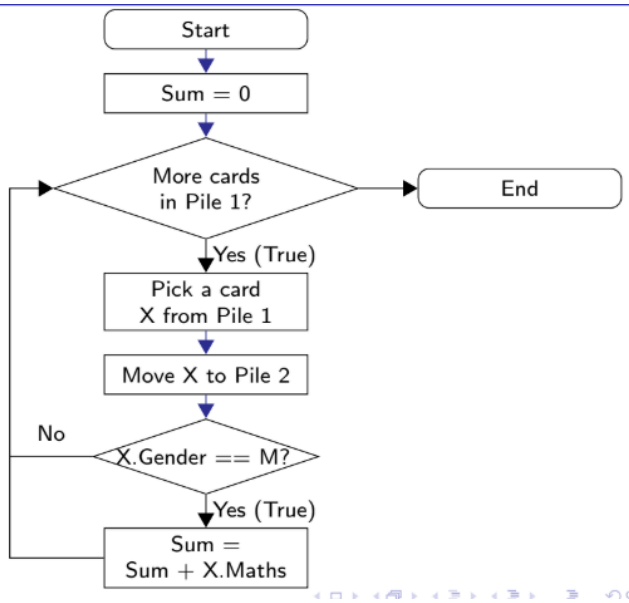
Lecture 6 : Pseudocode for procedures and parameters

Pseudocode: Procedures

Sum of Boys' Maths marks

```
Sum = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Gender == M) {
    Sum = Sum + X.Maths
  }
}
```

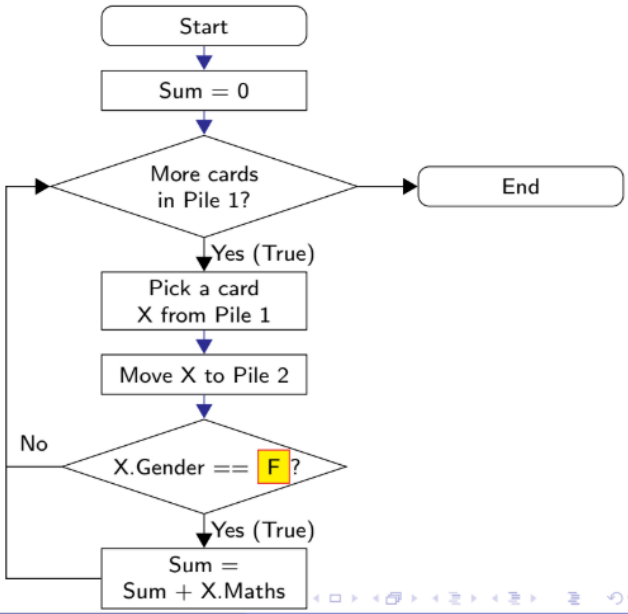
- What if we want to sum Maths marks of girls?



Sum of Girls' Maths marks

```
Sum = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Gender == F) {
    Sum = Sum + X.Maths
  }
}
```

- Only change is the value we check for **X**.Gender
- Remaining pseudocode is identical



A procedure to sum up Maths marks

- Procedure name: **SumMaths**
- Argument receives value: **gen**
- Call procedure with a parameter **SumMaths(F)**
- Argument variable is assigned parameter value
- Procedure call **SumMaths(F)**, implicitly starts with

  gen = F
- Procedure returns the value stored in **Sum**

```
Procedure SumMaths(gen)
    Sum = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.Gender == gen) {
        Sum = Sum + X.Maths
      }
    }
    return(Sum)
end SumMaths
```

A procedure to sum up Physics marks

- Only change is the field we examine in the card

  X.Physics, instead of **X**.Maths
- For Chemistry, add up **X**.Chemistry
- For Total, add up **X**.Total
- Pass field name as parameter

```
Procedure SumPhysics(gen)
    Sum = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.Gender == gen) {
        Sum = Sum + X.Physics
      }
    }
    return(Sum)
end SumPhysics
```

A procedure to sum up any type of marks

- Two parameters, gender (**gen**) and field (**fld**)
- **gen** is assigned a value, M or F, to check against **X**.gender
- **fld** is assigned a field name, to extract appropriate card entry **X**.fld
- Single procedure **SumMarks** to handle different requirements
  - **SumMarks(F,Chemistry)** Sum of Girls' Chemistry marks
  - **SumMarks(M,Physics)** Sum of Boys' Physics marks
  - ...

```
Procedure SumMarks(gen,fld)
    Sum = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

Calling a procedure

- Use procedure name like a math function, as part of an expression
- Assign the return value to a variable
- A procedure may not return a value
- Correct marks for one subject on a card
  - **Procedure UpdateMarks(CardId, Subject, Marks)**

```
GirlChemSum = SumMarks(F,Chemistry)
BoyChemSum = SumMarks(M,Chemistry)
if (GirlChemSum ≥ BoyChemSum) {
  "Congratulate the girls"
}
else {
  "Congratulate the boys"
}
```

- Use procedure name like a math function, as part of an expression
- Assign the return value to a variable
- A procedure may not return a value
- Correct marks for one subject on a card
  - **Procedure UpdateMarks(CardId, Subject, Marks)**
- Procedure call is a separate statement

```
GirlChemSum = SumMarks(F,Chemistry)
BoyChemSum = SumMarks(M,Chemistry)
if (GirlChemSum ≥ BoyChemSum) {
   "Congratulate the girls"
}
else {
   "Congratulate the boys"
}
```

```
Sum = 0
...
UpdateMarks(17,Physics,88)
...
GirlChemSum =SumMarks(F,Chemistry)
```

Summary

- Procedures are pseudocode templates that work in different situations
- Delegate work by calling a procedure with appropriate parameters
  - Parameter can be a value, or a field name
  - SumMarks(M,Total)
- Calling a procedure
  - Procedure call is an expression, assign return value to a variable
    - GirlsChemSum = SumMarks(F,Chemistry)
  - No useful return value, procedure call is a separate statement
    - UpdateMarks(17,Physics,88)
- Procedures help to modularize pseudocode
  - Avoid descibing the same process repeatedly
  - If we improve the code in a procedure, benefit automatically applies to all procedure calls

QN : 2,7

Lecture 7 : Pseudocode for procedures and parameters (Part 2)

Pseudocode: Using Procedures

Analysis of top students

- Is there a single student who is the best performer across subjects?
- Is the highest overall total the same as the sum of the highest marks in each subject?
- Need to compute maximum for different fields in a score card
  - Maths, Physics, Chemistry, Total
- Ideally suited to using procedures
  - Same computation with a parameter to indicate the field of interest

Finding the maximum in a given field

- As usual, keep track of the maximum using a variable
  - Initialize to 0
  - Update whenever you see a bigger value
- The value to be compared is not fixed
  - Parameter fld determines the field of interest

```
Procedure Maxmarks(fld)
    Maxval = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.fld > Maxval) {
        Maxval = X.fld
      }
    }
    return(Maxval)
end Maxmarks
```

Analysis of top students

- Use the procedure Maxmarks to find maximum marks in different categories
  - Four procedure calls, with fld set appropriately
  - Save each return value separately
- Use saved return values to compare the maximum overall total with the sum of the maximum subject totals

```
MaxMaths = Maxmarks(Maths)
MaxPhysics = Maxmarks(Physics)
MaxChemistry = Maxmarks(Chemistry)
MaxTotal = Maxmarks(Total)
SubjTotal = MaxMaths + MaxPhysics
            + MaxChemistry
if (MaxTotal == SubjTotal) {
  SingleTopper = True
}
else {
  SingleTopper = False
}
```

Summary

- Use a procedure when the same computation is used for different situations
  - Parameters fix the context
- Use variables to save values returned by procedures
  - Keep track of the outcomes of multiple procedure calls
- Procedures help to modularize pseudocode
  - Avoid descibing the same process repeatedly
  - If we improve the code in a procedure, benefit automatically applies to all procedure calls

QN : 2,3

Tutorial - 3.3

```
SumBB = 0
CountBB = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumBB, CountBB = AddIfBigBazaar ( X , SumBB , CountBB)
    Move X to Pile 2
}
AvgBB = SumBB / CountBB
```

```
Procedure AddIfBigBazaar (X, SumBB, CountBB)
    if (X.shopname == "Big Bazaar") {
        SumBB = SumBB + X.TotalBillamount .
        CountBB = CountBB + 1
    }
    return ([SumBB , CountBB])
    End addIfBigBazaar .
```

#1 Wrong Procedure Name.

#2 Wrong No of Parameters. ✓

#3 Incorrect Order of Parameters. ✓

Lecture 8 : Pseudocode for three prizes problem

Pseudocode: Awarding three prizes

Awarding three prizes

Want to award prizes to top 3 students

- Basic criterion is total marks
- Must also be within top three in at least one subject
- Must select at least one boy and one girl for top three prizes

Basic pattern

- Find top three marks in a category
- How is this to be done?

Finding maximum

- Initialize max to 0, scan cards, update each time you see a bigger value

Finding top two marks

- Maintain two values, max and secondmax
- If current card value is bigger than max
  - Copy max to secondmax, update max to current value
- If current card value is between max and secondmax
  - No change in max, update secondmax to current value

Top three marks in a subject

- Maintain max, secondmax, thirdmax
- Scan through all the cards
- Update max, secondmax, thirdmax as approprate

Procedure TopThreeMarks(Subj)

```
max = 0
secondmax = 0
thirdmax = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  ...
  ...
}
```

End TopThreeMarks

Case 1 :

- Maintain max, secondmax, thirdmax
- Scan through all the cards
- Update max, secondmax, thirdmax as approprate
- Current value is
  - Bigger than max

```
Procedure TopThreeMarks(Subj)
    max = 0
    secondmax = 0
    thirdmax = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      if (X.Subj > max) {
        thirdmax = secondmax
        secondmax = max
        max = X.Subj
      }
    }
End TopThreeMarks
```

Case 2 :

- Maintain max, secondmax, thirdmax
- Scan through all the cards
- Update max, secondmax, thirdmax as approprate
- Current value is
  - Bigger than max
  - Between max and secondmax

```
Procedure TopThreeMarks(Subj)
    max = 0
    secondmax = 0
    thirdmax = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      ...
      if (max > X.Subj > secondmax) {
        thirdmax = secondmax
        secondmax = X.Subj
      }
    }
End TopThreeMarks
```

Case 3 :

- Maintain max, secondmax, thirdmax
- Scan through all the cards
- Update max, secondmax, thirdmax as approprate
- Current value is
  - Bigger than max
  - Between max and secondmax
  - Between secondmax and thirdmax

```
Procedure TopThreeMarks(Subj)
    max = 0
    secondmax = 0
    thirdmax = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      ...
      ...
      if (secondmax > X.Subj > thirdmax) {
        thirdmax = X.Subj
      }
    }
End TopThreeMarks
```

- Maintain max, secondmax, thirdmax
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate
- Current value is
  - Bigger than max
  - Between max and secondmax
  - Between secondmax and thirdmax
- Need to return three values as a list $[v_1, v_2, v_3]$? Lists later ...

```
Procedure TopThreeMarks(Subj)
    max = 0
    secondmax = 0
    thirdmax = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      ...
      ...
    }
    return([max,secondmax,thirdmax])
End TopThreeMarks
```

- Maintain max, secondmax, thirdmax
- Scan through all the cards
- Update max, secondmax, thirdmax as approprate
- Current value is
  - Bigger than max
  - Between max and secondmax
  - Between secondmax and thirdmax
- Need to return three values as a list $[v_1, v_2, v_3]$? Lists later ...
- Sufficient to return thirdmax

Procedure TopThreeMarks(Subj)

max = 0
secondmax = 0
thirdmax = 0
while (Pile 1 has more cards) {
  Pick a card **X** from Pile 1
  ...
  ...
}
return(thirdmax)

End TopThreeMarks

Top three marks in a subject, in entirety

```
Procedure TopThreeMarks(Subj)
    max = 0
    secondmax = 0
    thirdmax = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      if (X.Subj > max) {
        thirdmax = secondmax
        secondmax = max
        max = X.Subj
      }
      if (max > X.Subj > secondmax) {
        thirdmax = secondmax
        secondmax = X.Subj
      }
      if (secondmax > X.Subj >
                          thirdmax) {
        thirdmax = X.subj
      }
    }
    return(thirdmax)
End TopThreeMarks
```

Three prizes

- Top three totals such that top three in at least one subject
  - Deal with boy/girl requirement later
- Again, maintain and update max, secondmax, thirdmax
- Scan through all the cards
- For each card, update max, secondmax, thirdmax as before
  - But only if in the top three of at least one subject!
  - Record third highest mark in each subject
  - Compare with subject marks before updating max, secondmax, thirdmax
- After scanning all cards, we have three prize winning totals
  - But who are the winners?
  - Keep track of card number of prize winners

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid

```
max = 0
secondmax = 0
thirdmax = 0

maxid = -1
secondmaxid = -1
thirdmaxid = -1
```

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject

⟨ Initialization of max, maxid etc ⟩

maths3 = TopThreeMarks(Maths)

phys3 = TopThreeMarks(Physics)

chem3 = TopThreeMarks(Chemistry)

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards

⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩

while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  ...
  ...
}

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate

⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩

while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  ...
  ...
}

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate
  - Only if top three in some subject — new procedure **SubjectTopper(...)**

⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩

while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  if (SubjectTopper(X,math3,phys3,chem3)){
    ...
    ...
  }
}

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate
  - Only if top three in some subject — new procedure **SubjectTopper(...)**

```
⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  if (SubjectTopper(X,math3,phys3,chem3)){
    if (X.Total > max) {
      thirdmax = secondmax
      thirdmaxid = secondmaxid
      secondmax = max
      secondmaxid = maxid
      max = X.Total
      maxid = X.Id
    }
  }
```


- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate
  - Only if top three in some subject — new procedure **SubjectTopper(...)**


```
⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  if (SubjectTopper(X,math3,phys3,chem3)){
    ...
    if (max > X.Total > secondmax) {
      thirdmax = secondmax
      thirdmaxid = secondmaxid
      secondmax = X.Total
      secondmaxid = X.Id
    }
  }
}
```

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate
  - Only if top three in some subject — new procedure **SubjectTopper(...)**

```
⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  if (SubjectTopper(X,math3,phys3,chem3)){
    ...
    ...
    if (secondmax > X.Total > thirdmax) {
      thirdmax = X.Total
      thirdmaxid = X.Id
    }
  }
}
```

- Maintain max, secondmax, thirdmax, as well as maxid, secondmaxid, thirdmaxid
- Record third highest mark in each subject
- Scan through all the cards
- Update max, secondmax, thirdmax as appropriate
  - Only if top three in some subject — new procedure **SubjectTopper(...)**
- In the end, we have what we need

```
⟨ Initialization of max, maxid etc ⟩
⟨ Record third highest per subject ⟩
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  ⟨ Update max, maxid etc ⟩
  }
}
```

Variables of interest

- maxid, max
- secondmaxid, secondmax
- thirdmaxid, thirdmax

Subject topper

- Compare each subject's marks on card with third highest
  - Passed explicitly as parameters
- One or more comparisons should succeed — **or** operator
- Value returned is a Boolean — True or False
- Typically, we would call this as follows:

  if (SubjectTopper(X,M,C,P) == True)
- Return value is Boolean, so ...

  if (SubjectTopper(X,M,C,P))

Procedure SubjectTopper
(Card,MMark,PMark,CMark)

```
if (Card.Maths ≥ MMark or
    Card.Physics ≥ PMark or
    Card.Chemistry ≥ CMark) {
    return(True)
}
else {
    return(False)
}
```

End SubjectTopper

Three prizes, in entirety

```
max = 0
secondmax = 0
thirdmax = 0
maxid = -1
secondmaxid = -1
thirdmaxid = -1
maths3 = TopThreeMarks(Maths)
phys3 = TopThreeMarks(Physics)
chem3 = TopThreeMarks(Chemistry)
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  if (SubjectTopper(X,math3,phys3,chem3)){
    if (X.Total > max) {
      thirdmax = secondmax
      thirdmaxid = secondmaxid
      secondmax = max
      secondmaxid = maxid
      max = X.Total
      maxid = X.Id
    }
    if (max > X.Total > secondmax) {
      thirdmax = secondmax
      thirdmaxid = secondmaxid
      secondmax = X.Total
      secondmaxid = X.Id
    }
    if (secondmax > X.Total > thirdmax) {
      thirdmax = X.Total
      thirdmaxid = X.Id
    }
  }
}
```

Boundary conditions

- What if all prize winners are of the same gender?
- Exclude the third prize winner and repeat the process
  - How many times?
  - Till we get three prize winners with at least one boy and one girls
  - Will this always given us three valid prize winners?
- What if there are ties?
  - How many ties can we tolerate?
  - Does it depend on first, second or third position?

Summary

- We have worked out a complex problem in full detail
- Identify natural units to convert into procedures
  - TopThreeMarks(Subj)
  - SubjectTopper(CardId,MMark,PMark,CMark)
- Shortcut for checking return value of a procedure that returns a Boolean value
  - if (SubjectTopper(CardID,Math3,Phys3,Chem3))
- Have to anticipate and account for unexpected situations in data
  - All toppers are same gender
  - Ties

QN : 1,4,5

Lecture 9 : Side effects of procedures

Pseudocode: Side effects

A procedure to sum up any type of marks

- Two parameters, gender (**gen**) and field (**fld**)
- What about the set of cards?
- This procedure works for a fixed set of cards
- Pass the deck of cards through a third parameter!

```
Procedure SumMarks(gen,fld)
    Sum = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes
  - Cards move from **Deck** to **SeenDeck**
- At the end of the procedure, **Deck** is empty!
- Procedure should also restore **Deck**
- Is this sufficient?

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**
  - **SeenDeck** would normally be in reverse order
- Side effect Procedure modifies some data during its computation

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal
  - Procedure to arrange cards in decreasing order of Total Marks
- A side effect could be undesirable
  - We pass a deck arranged in decreasing order of Total Marks
  - After the procedure, the deck is randomly rearranged

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity
  - Can the procedure have side effects?
  - Is the nature of the side effect predictable?
    - For instance, deck is reversed

Contract specifies interface

- Can change procedure implementation (code) provided interface is unaffected

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Side effects . . . Ex

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards
- Stop at the nearest name

Write a procedure for pronoun matching

- **FindMatch(before, pronoun, after)**
  - Three parameters
- **FindMatch** should not disturb **before** and **after**
  - Sequence of words, position
- No side effects should happen

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface
- Side effects are important to be aware of
  - Sometimes no guarantee is needed (adding up mark)
  - Sometimes no side effect is tolerated (pronoun matching)
  - Sometimes the side effect is the goal (sort the data)

QN : 3

PA : 3,9,10,11,12
GA : 3,4,6,8,9