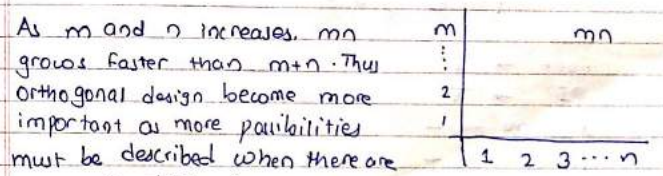
Q. Why does orthoganality simplify a language?

→ If we assigned an arbitary number to each pseudo-code operation, it would be necessary for the programmer to remember 20 independent facts for the 20 pseudo-code operations. Instead, we reflect in the coding the distinction between the direct and inverse operations. Therefore, structure in the operations is reflected in structure in the coding. The result is that the programmer only has to remember 12 independent facts; the plus (+) and minus (-) signs associated with the direct and inverse forms of the operation and the group that is associated with each of the digits. Another way to express this is the well-known architectural principle : form follows function

Orthogonal means right angled. If we have two independently meaningful axes one with m positions and another with n position. Then we can describe mn different possibilities even though we have to memorize m+n independent facts

As m and n increases, mn grows faster than m+n. Thus orthogonal design become more important as more possibilities must be described when there are many possibilities it may be advantage to have more than two orthogonal axis



Algol

$$X = \sum_{i=1}^{n} V_i$$

X := sum(i, 1, n, V[i])

```
begin real procedure sum(K, l, u, ak);
    value l, u;
    integer K, l, u;  real ak
    begin real s;
        s := 0;
        for K := l step 1 until u do
            s: s + ak;
        sum := s;
    end;
```

LISP

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```
(quotient (plus (minus b) (sqrt (difference
        (expt b 2) (times 4 a c))))
    (times 2 a))
```

LISP

Fact → !
PI → π
difference → −
plus → +
quotient → ÷
times → *
sqrt → √
expt → a^b

Chapter- 1 old qn

2017 Fall

(2/3) What are Amplicative and Reductive tools? Explain the phenemanology of programming Languages.

→ Amplicative: adding to (थप्नु)
Reductive: shrinking, subtractive to

'Technology utopians' tends to focus on the ampliative aspect - the increased reach and power - and to ignore the reductive aspect.
'Technology dystopians' tends to focus on the reductive aspect - the loss of direct, sensual experience - and to diminish the practical advantage of the tool
- we should acknowledge every tools have positive and negative aspects. As Ihde says, 'All technology is non-neutral'.

Example: Ihde contrasts the experience of using your hands to pick fruit with that of using stick to knock the fruit down
a. Stick is ampliative, it extends your reach to else inaccessible fruit.
b. Stick is reductive too, your experience of the fruit is mediated by stick, for you do not have experience of grasping the fruit and tugging (तान्ने) off the branch. You cannot feel if the fruit is ripe before you pick it

कुनै tools ले गर्दा केहि सुविधा थपेको हुन्छ भने त्यसले गर्दा केहि कमिपनी त्याएको हुन्छ।

In simplified terms, phenomenology means inner perception of reality. Phenomenology is the study of structures of consciousness as expressed from the P first person point of view. Fortunately, the phenomenology of tools that has been explored in detail, and in this section I will be using the results of investigation of Don Ihde

a. Tools are ~~addicti~~ Ampliative and Reductive

b. Fascination and Fear are common reactions to New Tools

c. With Mastery, Objectification Becomes Embodiment

d. Programming Language influence Focus and Actio

(Describe each of them)

11b] Do you think that it is necessary to learn principle programming Language for Software Engineering student What are the characters of good programming Langu

→ The direct answer is yes, It is necessary to learn principle of programming Language for Software engine students. Acquiring and developing knowledge about programming are a highly complex process. Programmi courses are regarded as difficult and often have very lo passing rates. Thats why, student should learn principle of programming Language. It helps them to better understand of prog. lang.

The main goals of principles of programming Language are Liited below:

- become familar with fundamental concepts of Computer Science
- Develop proficiency in an engineering problem solving and design methodology
- understand the importance of advanced Information technologies.
- Use computers and application software are tools to solve problems.

- The main reason is to study POPL is to provide the good knowledge about the fundamental to design, implementation, ~~of~~ and application of programing Languages.
- It supports to understand the programing lang Paradigms.
- It helps to understand the basic Logic of programming languages and its adherents and detractors.

- Improved background for choosing appropriate Language
- Provides greater ability to learn new languages
- Understand significance of implementation
- Ability to design new Language
- Overall advancement of computing

→ The characters of good programming Languages are enlisted. Many new hundreds of language are being designed and developed. There are many reasons to behind success or failure of good programing Language. Due to major importance of some of these external influence, it is the programmer who ultimately decides which language will survive and which will die!!!

Some of the important reasons are enlisted here.

a. Clarity, Simplicity and unity

A good programing Language must be simple and easy to learn and use. A PL provides a medium te execute our thought process into real coding statements such that primitives of language can be utilized to develop algorithms. It is desirable to have minimum number of different concepts, so that combining multiple concepts won't be that complex in nature.. It should be simple and regular as possible.

A complex syntax language may be easy to write program in, but it proves to be difficult to read and debug.

Ex: APL programs are complicated so that even developers find it difficult to understood after 1-2 months. however, power needed for language should not be scarcificed for simplicity.

b. Orthogonality: (कुछ features छ हैं, कुछलाई combine गर्दा काम गर्ने पर्‍यो ie if and conditional expression

- It refers the attribute of being able to combine various features of a language in all possible combinations, with every combinations meaningful.
- Orthogonality help to develop many new algorithms.

Ex: Suppose language provides an arithmetical calculation operator, another conditional statement, which o/p are either 0 or 1, Now the language should support combinations of these two expression.

c. Naturalness:

A good language should be natural for the application area for which it is designed. That is, it should provide appropriate operators, data types, control structure and

Ex: plate & गर्ने, Stack data type हुन पर्‍यो

natural syntax to facilitate programmers to code their problems easily and efficiently. ie- FORTRAN is good example.

d. Support for Abstraction

Abstraction means ability to define and then use complicated structures or operations in ways that allow many of the details to be ignored. Many times language fail to implement many real life problems into programs

Ex: c++ is one of the most used language, that provides such facilates

Consider a situation where a scheduling is to be done for college student for attending a lecture in a class section, teacher, It is not provided by C, but c++ has it.

e. Ease of Program Verification:

programs verification should be provided by language to check and minimize the errors. Sometimes testing the program with random values of the inputs and obtaining corresponding outputs.

f. Programming environment:

environment play a vital role in success of language. The environment (IDE) which is technically weak may get a bad response of programer. Environment itself more ease of PL.

g. Portability of programs

Transportability of resulting programs from one computer to other systems. A language which is widely available and doesnot support features on different computer system. (hardware मा Independent हुन पर्‍यो)

8. Cost of use:
here cost is time, resource:
- cost of program execution
- cost of program Translation
- cost of program Creation, Testing and use
- Cost of Program Maintainance

9. Compactness
10. Extensibility

* 2016 Spring

[10] Compare and Contrast pseudocode Interpreter, Symbolic pseudo-code Interpreter and an assembler.

→ pseudocode is an instruction code that it different from that provided by the machine - preassumbly better. The design of an "interpretive subroutine" for executing the pseudocode and design we call it an Interpreter. They present interpreters primarily as a means of saving memory since pseudocode more compact that machine real instruction code.
The significance of these pseudo code

[16] Explain different aspects of designing of Pseudocode programming language.

→ • Before designing the Pseudocode, it's basic capabilities must be decided. So for this lets assumes that we are designing Pseudocode for a computer with 2000 words of 10 digit memory; of course we want our virtual computer to provide the facilities found in any computer, such as arithmetic, control of execution flow and input-output. So lets begin by making a list of some of the functions:
+ Floating-point arithmetic (+, -, *, /)
+ Floating-point comparisions (=, ≠, <, >, ≤, ≥)
+ Indexing
+ Transfer of control
+ Input-output

• what Syntax should be used for this Psuedocode? Another aspect is desiging Syntax, since earlier language doesnot include alphabetic input/output
• Another is address, since many language use 1 sign bit and other as address. ~~pseudocode~~
• Another aspect is designing Interpreter itself
• handling security
• Indexing and Loops
Pseudocode provides built in Indexing. To provide Indexing, we need the address of the array and array of index variable. Therefore, the only operations we can perform directly on array elements are to move them to or from other locations. we can use the code +6 and -6 to move from or to an array
Example: if there is 100 element array begining at location 050 in data memory, and location 050 contains 17

then
+6 250 080 803
will move the contents of location 297 (=280+17) to location 803

• One of the main reasons of using array is that we can a loop to ~~person~~ perform the same operation on each element of array. To do this ~~can~~ require us to be able to initialize, increment, and test index variable. we may expect arithmetic and comparison facilities.

• A program is not useful, if it can't read data or print a result. Pseudocode will use +8 operation to read a card containing one 10 digit number into specified memory location and -8 operation to print the contents of memory location. In real pseudocode, a punch operation could be more common than a print operation

[PPL - IC Fall]

[4a] What are characteristics of good programming language? Why is it important for software engineers to study principles of programming languages? Explain

→ Many new hundreds of programming language were developed and very few are sustained for 10 years. There are many reasons behind success or failure of language. Sometimes, the reason may be external to the language itself. It is the programmer who ultimately decides which language will survive and which will die The following important reasons are enlisted here:

1. Clarity simplicity and Unity

A programming language provides conceptual thinking ie execute thought into real coding statement. for algorithm to be implemented on language, its need is that the language should be quite clear, simple and unified. It is desirable to have minimum different concepts. Combining multiple concepts ~~can't~~ be that complex in nature. The main concern of language is readability. A complex syntax language may be easy to write a program in, but difficult to read and debug.

2. Orthogonality:

Attribute of being able to combing various features of a language in all possible combination with every combination being meaningful. Eg: IF Statement and Conditional statement must be combine.

3. Naturalness for the Application

A language needs a syntax that, when applied properly allows the program structure to reflect the logical structure what a programmer wanted it to. The language should provide appropriate data structures operations, control structures and a natural syntax for the problem: eg: STACK structure for plate problem

4. Support for Abstraction:

Many times prog lang fail to implement many real life problem. There is always a gap bet° abstract data structure and operations. Eg: For scheduling of college student for attending a lecture in a class section. C language fail to assign a student a section lecture and teacher to attend. so, C++ required

5. Ease of Program verification

The reliability of prog lang. written in a language is always a central concern. There are many techniques which can be used to keep track of correct functionality of a language. Sometimes testing the program with random values of input and obtaining corresponding outputs. Program verification should be provided by language to check & minimize the errors.

6. Programming environment.

The env. which is technically weak may get a bad response as programmer rather than a language that has less facility than the former but its environment is technically good. Special editors, testing packages tailored to the language.

7. Portability of programs

The resulting program must be transport from one computer to another and must be hardware independent Ex: C, C++

8. Cost of use

program execution costs is the total amount which has been used to implement the program. The research work on design, optimizing compilers, data allocation requires etc

- Cost of program translation
- Cost of program creation, Testing and use
- Cost of program maintainance.

9. Well Documentation

10. GUI of language

Second part.

→ Programming language is a language that is intended for the expression of computer programs and that is capable of expressing any computer programs. The principle of programming language comes from concatenating principle + programming language. principle of programming language is a set of rules and norms governed to ~~Compo~~ communicate instructions to a machine or particularly a computers. The study of principle of programming language supports to better understands the basic logics of programming languages and its adherent and detractor. It support to understands the language paradigm. The potential benefits for software engineers to study it are:

a. Increased ability to express ideas :

- It is widely believed that the depth at which a programmer think is influenced by the expressive power of the language in which programmer communicate our thoughts. It is difficult for people to conceptualize structures they can't describe, verbally or in writing.

b. Improved background for choosing appropriate language

- By studying, popL, programmer can choose best suitable from the available lang. Many programmers, when given a choice of language for a new project, continue to use the language with which they are familiar even if it is poorly suited to new projects
- If these soft engr were familiar with other languages, they would be in better position to more informed lang. choices

c. Provides greater ability to learn new language :

- programmers who understand oo concept can learn Java easily
- Once a through understanding of fundamental concepts of lang is acquired, it becomes easier to see how concepts are incorporated into the design of lang: being learned

d. Understand Significance of Implementation
- understanding of implementation issue leads to an understanding of why languages are designed the way they are so that we can use language more intelligently

e. Ability to design new language:
The more languages you gain knowledge of, the better understanding of programming languages concepts you understand.

f. Overall Advancement of computing
Finally, there is a global view of computing that can justify the study of programming language concepts. Although it is usually posible to determine why a particular prog lang (programming language) became popular. A language became widely used at least in part.

[11b] What are the major programming language domain? Explain the application of pseudocode in programming.

→ Programming domain defines the ability of using a specific language for specific usage

1. Scientific Application
Scientific application typically need simple data structures and arrays and many floating point operation and performs matrix manipulation. Java is not proper lang for this. C should be used for scientific application

2. Business Application
we need to able to represent data in a human understandable way ie chart, bar

3. Artificial Intelligence

4. System programming

5. Web services

The term pseudocode is often now used for informal program design notations; which are not intended to be executable by a computer hence pseudo. However; pseudo-code is used in its orignal senses: primitive, interpeted programming language.
The application of pseudocode ~~is~~ are:

a. Floating point and Indexing were simulated
b. Interpreter is designed for executing pseudo-code
c. Virtual computer over real computer provides the ~~m~~ facilities more suitable to application and eliminate many details from the programmer
d. Indexing and loops was easier
e. Input-output function were added

The pseudocode is an informal way of programming description that doesnot require any strict programming language syntax or underlying technology considerations.
- It is understood by all programmers of all types
- It enables the programmer

[2015 Spring]

[11a] Do you think PODL is essential to software Engineering Students and why? what are the major four domains of programming language.

Yes, I think PODL is essential because:
a. Increased capacity to express ideas.
b. Improved ability to choose appropriate languages
c. ~~keep~~ helps to learn new language
d. Understand significance of implementation
e. Improved use of language one already "knows"

The application domain are:

1. Scientific Application
2. Business Application
3. Artificial Application
4. web Application

[11b] What is Pseudocode and pseudocode Interpreter? Explain about the functional enhancement brought by Pseudocode ~~by Pseudo ca~~ during 1st gen of ~~Prog lang~~.

→ Pseudocode is a first generation programming language which is intended to overcome the earlier programming issues like floating point and indexing. It is primitive and interpreted programming language.

Pseudocode Interpreter is a tools to write and execute the pseudo-code. pseudocode interpreter ~~is~~ were implement a virtual computer with its own set of ~~data~~ data types and operation.

The functional enhancement are:

- floating-point arithmetic ($+, -, \times, \div, \sqrt{}$)
- floating point comparision ($=, \neq, <, >, \leq, \geq$)
- Indexing
- Transfer of control
- Input output.

[14 spring]

[10] What are different areas and capabilities that you will consider while designing new programming language and why?

→ All programming language can perform equivalent tasks as each other. Some language can solve certain types of problems much easier than other language Eg: easier to use LISP or prolog to manipulate symbols and logics. Different language are used for different purposes.

- The closer a language can solve a particular problem, the more likely it will gain acceptance when devs need to solve particular problem. language without proper streghth is unlikely to gain adoptation, is similar to starting business - you have to have an unique selling proposition

- A new language must have basic things such as- robustness, performance, documentation, tons of libraries, healthy community etc
- The popular term today is DSL (Domain specific Language) and main point that your lang should solve one problem well above and beyond the others. Some DSL's are
  - SQL (database language)
  - PHP (web processing)
  - R (statistical language)
  - make (build dependency mgmt)
  - Shell Scripting (automate os)
  - PostScript (lang for Printer instruction)
- Inorder to design new language, we must have compelling solution against others. we should know how other solve the same problem currently. we should know for 2 reasons
  1. Know what they do WELL - borrow these
  2. Know what they do POORLY - have better solutions
- from language design perspective, the easiest easy to do are
  1. baking in what a library level feature into lang. itself
  2. Simplify the syntax.

- Creating the language is one of the bigger 'reinvent the wheel'. The easy way to leverage existing knowledge of the users. If we are designing new language, we will make language look somewhat like C than to look like Haskell coz more developers are attracted to C.
- We should make extensive documentation from the very begining so users can will need as little of it from very begining.
- Create code repository or community to share their code or to discuss the problem they are facing.

The features likely to have the basics -
- primitives data structure, control statements, operators, and functions
- It must have good environment (IDE), debugger

[1'b] "Programming language influences focus and action" justify

Tools influence the style of a project. For example, Inde contrast three writing technique : a dip pen, an electric typewriter and a word processor. In the case of a dip pen the speed of writing is so much slower than the speed of thought that a sentence can be crafted word by word as it is written. With typewriter, the speed of writing is closer to speed of thoughts. So this tools inclines towards a more informal styles. However revision require retyping. Incontrast with a word processor text can be edit in small units

In General, a tools influences focus and action. It influences focus by making some aspects of the situation salient and by hiding others. It also influence actions by making some easy and other awkwards. Like other tools, programming language influence the focus and actions of programmers and therefore their programming style.

A programming language inclines programmers toward a style; it creates a tendency, which the majority of programmers will follow. However, I must emphasize that it doesn't dictate a style; Individual programmers may choose to work against the language's inclination.

Example: we sometime observe a programmer "writing FORTRAN in LISP" ie writing FORTRAN-style code in the LISP language. Nevertheless, we must consider carefully the stylistic inclination of Programming language.

Does it encourage the focus and action that we want to incorage ?

[PPL-14 Fall]

[1'a] Define Programming language. What is the important of studying Programming language for Software Manager, Language designers and Implementers.

Programming language is a (computer) language that is intended for the expression of computer programs and that is capable of expressing any computer programs. It is a special language programmer use to develop software program, scripts or set of instructions for computer to execute. Each programming language has a unique set of keywords and special syntax for organizing program

- Machine level language / Low level
- Assembly language
- High-level language

Importance of programming language for

a. Software Managers
- Software mangers often makes decision regarding choosing the language (new or existing). for this, Software manage must know what common language can do cannot do
- Cost of designing or extending a project
- Cost of implementing, benefits of various language

b. Language Designers
- "Those who cannot remember the past are condemned to repeat it", As George Santayna said; An understanding of history makes you realize what previous language makes failure so that you can overcome it. language
- language designer must know very various types of language so that, a language designer can add the basic yet necessary features to new language.

c. Language Implementers
- If we are interested in language implementation, we will gain insight into the motivations for various language facilities thus allowing us to make reasonable implementation trade offs.
- Although, language implementation is complicated subject, so that we must have knowledge of different language for example: having knowledge of CSS must require the knowledge of HTML.

[1b] Discuss the phenomenology of programming language.
→ Already done [2017 Fall 1b]

1. Tools are Ampliative and Reductive.
2. Fascination and fear are common reaction to New Tools.
3. With Mastery, Objectification becomes Embodiment.
4. Programming language influences focus and Action.

Summarization
Programming language transform the situation encountered in programming projects. They are non neutral and have ampliative and reductive aspects, both of which should kept in mind. further, to assess the benefits and limitations of a programming language properly, it is necessary to advance beyond the fascination-fear stage. When a well designed language is mastered, it becomes a transparent extension of the programmer rather than an obtrusive object. Finally, by influencing the focus and actions of programmer a language inclines it user toward a particular style but it doesnot force it on them.

[PPL - 13 Spring]
1a and 1b are done previously

[PPL - 13 Fall]
1a) Any program that can be written in one also be written in another. Why, then, are there so many programming language? Explain
→ In a sense, Any program that written in one can also written in another. like A program written in C++, can be converted into java, C#, Kotlin Underlying this fact is that ~~as~~ many langeage on same domain serve the same purpose: to turn human thoughts into computer readable form 1's & 0's.

At their most fundamental level, these language are all the same. But on the surface - where humans interact with them - they vary alot. This is where other concerns come into play.

· Different Tools for different jobs:

programming language are tools, we choose different for different jobs. A tractor, bicycle and tesla are all vehicle, they have wheels, steering but obviously we use them for different thing. Programming languages are similar. Ruby and Javascript are great for websites; Java, C++ are often used for financial trading; Python and R for analysis statistics.

· Developers have tastes:

Beyond mere utility, developers choose tools based on personal tastes. A programming language is a tools for humans to express ideas to computer. while developers have many things in common, there is natural variety in the way our minds work. Because we have many choices of good programming languages, we can select one that "works the way I think". Some developers like Ruby's flexibility, while others prefer Java's strictness.

· People first:

Beyond utility, and beyond taste, business run on people. Often, you will choose a programming language based on what you, or the people around you, know. Stackoverflow uses C# mostly because that what's founders knew.

· Variety is strength:

We have a variety of programming language because there is a variety of jobs to be done and a variety of people who do those jobs.

New Programming language often learn from existing languages and add, remove and combine features in a new way.

· Some programming languages are targetted at specific problem domain ex: R for analysis, C for OS kernels.

Each and every programming language has its own priority and usage. every ~~language~~ human language can be translated to each other, eventhough there are many Programming language.

lol 5 marks को २६! डाला

1b] "Orthogonal design Increases Regularity". Justify

Orthogonality means that a relatively small set of primitives constructs can be combined in a relatively small number of ways to build control and data structures. Orthogonality is a relationship between two things such that they have minimal effect on each other. When we combine two or more primitives, it must make a sense.

Regularity is a measure of how well a language ~~interacts~~ integrates its features, so that there are no unusual restrictions, interactions or behaviour. Easy to remember.