MACHAKOS UNIVERSITY

UNIVERSITY EXAMINATIONS

SUPPLIMENTARYEXAMINATION FOR THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE

SCO 306: PROGRAMMING LANGUAGES

Date: **Time:**

Attempt Question ONE and ANY other TWO

Marking Scheme

QUESTION ONE

a. List the main FIVE goals (reasons) for studying the formal semantics of programming languages **[5 marks]**

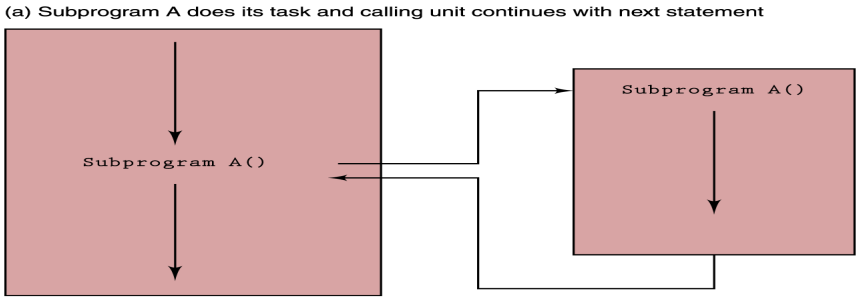
Answer:

i. Standardizing language definitions
ii. Proving properties about programs
iii. Assisting in language design
iv. Animation and language prototyping
v. Applications to compiler techniques

b. Using an example, explain how *sub routines* implement *flow of control* in programming language. [4marks]

Answer:

When the named function call is encountered, the processing in the other part of the program halts and the control is passed to the function for execution. The control is later returned back.



c. Discuss the following criteria used to determine the quality of programming language.

[6marks]

i. Extensibility
ii. Standardability
iii. Implementability

Answer:

i. **Extensibility:** The quality of a language that provides some general mechanism for the user to add new constructs to a language.
ii. **Standardability:** The quality of a language that allows programs written to be transported from one computer to another without significant change in language structure.
iii. **Implementability:** The quality of a language that provides a translator or interpreter can be written. This can address to complexity of the language definition.

d. List any FIVE Primary Reasons for Good Programming Language Design **[5 marks]**

Answer:

i. Increased ability to express ideas
ii. Improved background for choosing appropriate languages
iii. Increased ability to learn new languages
iv. Better understanding of significance of implementation
v. Better use of languages that are already known
vi. Overall advancement of computing

QUESTION TWO

a. Object-oriented programming is one of the programming language that is in wide use today. Explain the following terms as used in Object Oriented programming **[10 marks]**
   i. Polymorphism
   ii. Abstraction
   iii. Message passing
   iv. Encapsulation
   v. Class

Answer:

i. **Polymorphism**: - ability is objects within the same class to take different forms
ii. **Abstraction**: - Abstraction is simplifying complex reality by modeling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem.

iii. **Message passing**: - The process by which an object sends data to another object or asks the other object to invoke a method Also known to some programming languages as interfacing.

iv. **Encapsulation**: - Encapsulation conceals the functional details of a class from objects that send messages to it.

v. **Class**: - Defines the abstract characteristics of a thing (object), including the thing's characteristics (its attributes, fields or properties) and the thing's behaviors (the things it can do, or methods, operations or features). Or class is a blueprint or factory that describes the nature of something.

b. Explain the principle of functional languages. Give two examples of functional languages. **[5marks]**

Answer:

A program in this paradigm consists of *functions* and uses functions in a similar way as used in mathematics

Program execution involves functions calling each other and returning results.

There are no variables in functional languages.

- Example functional languages include: ML, Miranda™, Haskell

c. Discuss the key principles of object oriented programming paradigm. **[5marks]**

Answer:

- Focus is on writing good classes (or types) that define operations on objects of the class
- : Object- Instance of a class. Has an associated state.
- Incorporates both encapsulation and inheritance through the class concept
- Data Abstraction (specifies behavior)
- Encapsulation (controls visibility of names)
- Polymorphism (accommodates various implementations)
- Inheritance (facilitates code reuse)
- Modularity (relates to unit of compilation)

QUESTION THREE

a. Explain the concept *Orthogonality* of a programming language. Give three ways in which a language may fail to achieve *Orthogonality* characteristic. **[6 marks]**

Answer:

- For a programming language to be orthogonal, language constructs should not behave differently in different contexts.

Give three ways in which a language may fail to achieve *Orthogonality* characteristic.

- In Pascal functions can only return scalar values or pointers.
- In C/C++, arrays types cannot be returned from a function
- In C, local variables must be at the beginning of a block.
- C passes ALL parameters by value except arrays (passed by reference).
- When expressions may not include function calls can be viewed as a non-orthogonality.

b. List and explain the TWO types of semantics in programming **[4 marks]**

Answer:

i. Static semantics - semantics which can be enforced at compile-time.

ii. Dynamic semantics - semantics which express the run-time meaning of programs.

c. List and explain the three major classes to formal semantics as used in programming languages **[6 marks]**

Answer:

i. Denotational semantics – this is whereby each phrase in the language is interpreted as a *denotation (meaning)*, i.e. a conceptual meaning that can be thought of abstractly. Such denotations are often mathematical objects inhabiting a mathematical space, but it is not a requirement that they should be so.

ii. Operational semantics – Operational semantics may define an abstract machine and give meaning to phrases by describing the transitions they induce on states of the machine. The execution of the language is described directly (rather than by translation).

iii. Axiomatic semantics - whereby one gives meaning to phrases by describing the *logical axioms* (sayings) that apply to them. Axiomatic semantics makes no distinction between phrase's meaning and the logical formulas that describe it; its meaning *is* exactly what can be proven about it in some logic.

d. Using examples differentiate between *concrete syntax* **and** *abstract syntax.* **[4marks]**

Answer:

Concrete syntax

Rules for writing expressions, statements, programs. Describes its written representation, including lexical details such as the placement of keywords and punctuation marks

Abstract syntax

Internal representation of expressions and statements, capturing their "meaning" (i.e., semantics)

Capture intent, independent of notation.

QUESTION FOUR

a. Discuss the following variations of formal semantics as applied in many programing languages **[8 marks]**

i. Action semantics
ii. Predicate transformer semantics
iii. Algebraic semantics
iv. Concurrency semantics

Answer:

i. Action semantics - this describes an approach that tries to modularize denotational semantics, splitting the formalization process in two layers and predefining three semantic entities (actions, data and outputs) to simplify the specification;
ii. Predicate transformer semantics - describes the meaning of a program fragment as the function transforming a post condition to the precondition needed to establish it.
iii. Algebraic semantics - this describes a form of axiomatic semantics based on algebraic laws for describing and reasoning about program semantics in a formal manner.
iv. Concurrency semantics - this describes a catch-all term for any formal semantics that describes concurrent computations. Historically important concurrent formalisms have included the Actor models

b. To manage the size of program code in many programming languages, control structures are mandatory. List and Explain the three control structures **[6 marks]**

Answer:

ii. Sequence - ordered statements or subroutines executed in sequence one after the other.
iii. Selection - one or a number of statements is executed depending on the state of the program. This is usually expressed with keywords such as <u>if…then…else...end if</u>
iv. Iteration - a statement or block is executed until the program reaches a certain state, or operations have been applied to every element of a collection. This is usually expressed with keywords such as <u>while</u>, <u>repeat</u>, <u>for</u> or <u>do..until</u>.

c. In context of *data integrity* and *flexibility*, compare and contrast *typed* and *untyped languages*. Give example language in each category. **[6marks]**

Answer:

- *Typed languages – good in data integrity, poor in flexibility – e.g Java, C++*
- *Untyped languages – poor in data integrity, good in flexibility –e.g PHP, Javascript.*

QUESTION FIVE

a. Explain the following types of data structures as its applied in many programming languages **[10 marks]**

i. Array
ii. Union
iii. Graphs
iv. Structures
v. Set

Answer:

i. Array - An array is a data structure whereby a number of elements in a specific order, typically all of the same type are stored in the same structure. Elements are accessed using an integer index to specify which element is required.
ii. Union - A union type specifies which of a number of permitted primitive types may be stored in its instances, e.g. *float* or *long integer*.
iii. Graphs and trees - are linked abstract data structures composed of *nodes*. Each node contains a value and one or more pointers to other nodes arranged in a hierarchy.
iv. Record (Structures) - A record (also called a *tuple* or *struct*) is an aggregate data structure. A record is a value that contains other values, typically in fixed number and sequence and typically indexed by names. The elements of records are usually called *fields* or *members*.
v. Set - A set is an abstract data structure that can store specific values, in no particular order and with no duplicate values.

b. In context of type systems, explain the following concepts. Use examples programming languages **[6mark]**

i. Type loopholes
ii. Latent typing
iii. Type-inferred

Answer:

i. Type loopholes - Allow one data type to be treated like another.
ii. *latent typing - determines the type-safety of operations at runtime; i.e, types are associated with runtime values rather than textual expressions*
iii. ***type-inferred*** - the compiler *infers* the types of expressions and declarations based on context. E.g . Var Name; Var number; Dim Salary; Dim EndDate

c. List any FOUR programming Language Evaluation Criteria when choosing the language to use in program development **[4 marks]**

Answer:

ii. Readability - the ease with which programs can be read and understood
iii. Writability - the ease with which a language can be used to create programs
iv. Reliability - conformance to specifications (i.e., performs to its specifications)
v. Cost - the ultimate total cost of the language