Introduction to Programming Languages

Programming in C, C++, Scheme, Prolog, C#, and SOA

The Third Edition (Revised Print)

Answer Keys to Multiple Choice Questions

Yinong Chen and Wei-Tek Tsai
Arizona State University
1.7  Homework and programming exercises

1. Multiple Choice. Choose only one answer in each question. Choose the best answer if more than one answer are acceptable.

1.1  □ imperative   □ object-oriented   □ functional   □ logic

1.2  □ imperative programming   □ functional programming

If you teach someone to order a pizza from a restaurant, you are in fact teaching
 □ imperative programming   □ functional programming

1.3  □ efficiency   □ orthogonality   □ reliability   □ readability

1.4  □ readability   □ writeability   □ efficiency   □ None

1.5  □ reduce the types of control structures.   □ increase the types of control structures.
 □ make programs to execute faster.   □ use BNF to define the syntactic structure.

1.6  □ Ada   □ C   □ Java   □ All of them

1.7  □ lexical   □ syntactic   □ contextual   □ semantic

1.8  □ how to form lexical units from characters.
 □ how to put lexical units together to form statements.
 □ the static semantics that will be checked by the compiler.
 □ the dynamic semantics of programs under execution.

1.9  □ the source program is small.
 □ the source program is written in an assembly language.
 □ the difference between source and destination is small.
 □ multi-module programming is used.

1.10 □ There is no difference between them.
 □ The *inline* functions are for Java only. There are no inline functions in C++.
 □ *inlining* is a suggestion to the compiler, while a *macro* definition will be enforced.
 □ A *macro* definition is a suggestion to the compiler, while *inlining* will be enforced.
2.10 Homework, programming exercises, and projects

1. Multiple Choice. Choose only one answer in each question. Choose the best answer if more than one answer are acceptable.

1.1
☐ provides a level of abstraction. ☐ is never necessary.
☐ is not required if <iostream> is included. ☐ is useless.

1.2
☐ C is not designed to handle logic operations.
☐ C uses strong type checking.
☐ Boolean values can be represented as integers.
☐ C++ already defined a Boolean type.

1.3
☐ one function is defined within the other function.
☐ they call each other.
☐ each function calls itself.
☐ they are independent of each other.

1.4
☐ 1
☐ 5
☐ 6
☐ 7
☐ 40

1.5
☐ str2 = str1; ☐ str2 = 0;
☐ str1 = str1+1; ☐ *str2 = "world";

1.6
☐ char s[5];
☐ char s[3] = "hello";
☐ char s[];
☐ char s[] = {'s', 't', 'r'};

1.7 Given a declaration: int i = 25, *j = &i, **k = &j;
which of the following operations will change the value of variable i?
☐ j++; ☐ k++; ☐ (*k)++;
☐ (**k)++;

1.8
☐ i++; ☐ (&i)++;
☐ (*j)++;
☐ (**k)++;

1.9
☐ call-by-value
☐ call-by-alias
☐ call-by-address
☐ None of them

1.10
☐ call-by-value
☐ call-by-alias
☐ call-by-address
☐ None of them

1.11
☐ call-by-value
☐ call-by-alias
☐ call-by-address
☐ None of them

1.12
1.13
☐ head-recursion  ☐ middle-recursion  ☐ tail-recursion  ☐ double recursion

1.14
☐ (s, t)  ☐ A(s, t)  ☐ A(n)  ☐ n

1.15
☐ s = 0 and t+1  ☐ s = 0 and t = 1  ☐ s > 0 and t = 0  ☐ s > 0 and t > 0

3.8  Homework, programming exercises, and projects

1. Multiple Choice. Choose only one answer in each question. Choose the best answer if more than one answer are acceptable.

1.1
☐ a pointer to a class A object can point to a class B object.
☐ a pointer to class B object can point to a class A object.
☐ a pointer to class A object can point to a class B object, and vice versa.
☐ a pointer to class A object can NOT point to a class B object, and vice versa.

1.2
☐ is an abstract interface that has no implementation.
☐ is an extendable function that allows programmer to add formal parameters to the function.
☐ implies early binding between the function name and the code.
☐ implies late binding between the function name and the code.

1.3
☐ any members of the derived class object.
☐ the inherited members of the derived class object.
☐ the additional members of the derived class object.
☐ public members in the derived class object.

1.4
☐ heap object created in the constructor.
☐ heap object created in main() function.
☐ stack object created in constructor.
☐ stack object created in main() function.

1.5
☐ heap object created in the constructor
☐ heap object created in main() function
☐ stack object created in constructor
☐ stack object created in main() function

1.6
☐ Only in A.
☐ Only in B.
☐ In A and B.
☐ None of them.
1.7
- $p = q$
- $q = p$
- Both of them.
- None of them.

1.8
- cannot be defined in class $A$.
- cannot be re-defined in a derived class.
- can be re-defined in a derived class.
- none of them.

1.9
- any members of the derived class object.
- the inherited members of the derived class object.
- the new (extended) members of the derived class object.
- public members in the derived class object.

1.10
- derive one class from the other (use inheritance).
- contain one class in the other.
- define them totally independent of each other.
- none of them.

1.11
- number of data fields in classes.
- number of member functions in classes.
- number of public members in classes.
- inheritance relationship among classes.

1.12
- zero or more.
- one only.
- one or more.
- at most two.

1.13
- exit from try-block and pass a value to a catch-block
- exit from catch-block and pass a value to the try-block
- same as try
- None of them.

1.14
- manager1
- manager2
- both of them
- none of them

1.15
- m.id
- m.empl.id
- m.empl->id

1.16
- n.id
- n.empl.id
- n.empl->id

1.17
- A type that can take different type of values at the same time.
- A type that contain all other types.
- The type can be determined at run time.

1.18
1.19
☐ To improve the performance of multithreading.
☐ To make sure that all the threads complete their tasks.
☐ To instantiate a concrete type to a generic type.

1.20
☐ defining generic type ☐ defining generic class
☐ defining parallel computing process ☐ saving memory usage

4.11  Homework, programming exercises, and projects

1. Multiple Choice. Choose only one answer in each question. Choose the best answer if more than one answer are acceptable.

1.1
☐ #t ☐ #f ☐ A ☐ error message

1.2
☐ #f ☐ (3 5 2 9) ☐ (4 2 1) ☐ (2 1)

1.3
☐ (6 8 10) ☐ (6) ☐ 6 ☐ (D) error message

1.4
☐ (NULL? L) ☐ (= (length L) 0)
☐ (< (length L) 1) ☐ (= L 0)

1.5
☐ (+ z 3) ☐ ((lambda (z) (+ z 3)) 4)
☐ (define foo (lambda (z) (+ z 3))) ☐ (define bar 25)
☐ none of them

1.6
☐ all parameters of a function first.
☐ a parameter of a function only if it is necessary.
☐ no parameters at all.
☐ outermost first.
1.7
- all parameters of a function first.
- a parameter of a function only if it is necessary.
- no parameters at all.
- innermost first.

1.8
- may produce different results.
- never produce different results.
- always produce different results.
- None of them are correct.

1.9
- may produce different results.
- never produce different results.
- always produce different results.
- None of them are correct.

1.10
- an unnamed procedure.
- a list of local variables.
- a list of global variables.
- a named procedure.

1.11
- (cons x y)
- (list x y)
- (append x y)
- None of them

1.12
- '()
- '(x . y)
- '(x)
- '(())

1.13
- (30 80)
- (10 15 10)
- (10 15)
- (10 10 15)

1.14
- is a plain list.
- contains sub lists.
- contains non-numerical values.
- is not a pair.

1.15
- 20
- 40
- 25
- (20 40 25)

1.16
- a pair.
- not a pair.
- a string.
- 0

1.17
- call-by-value
- call-by-reference
- call-by-name
- return value

1.18
- call-by-value
- call-by-reference
- call-by-name
- return value
1.19
- Use multiple return-statements.
- Split the procedure into multiple procedures.
- Put the values in a pair and return the pair.
- Put the values in a list and return the list.

1.20
- find the solutions of size-1, size-2, ..., size-n problems.
- find a loop variable that is incremented in each iteration.
- find the solutions of size-n, size-(n-1), size-(n-2), ..., size-1 problems.
- find the solution of size-(n-1) problem, and finally find the solution of size-n problem.
- find the solution of size-n problem based on the hypothetical solution of size-(n-1) problem.

5.9 Homework, programming exercises, and projects

1. Multiple Choice. Choose only one answer in each question. Choose the best answer if more than one answer are acceptable.

1.1
- C
- Lisp
- Scheme
- Prolog

1.2
- Lambda calculus
- Predicate logic
- Turing machine
- Boolean logic

1.3
- A fact can be considered a special case of a rule.
- A fact can be considered a special case of a goal.
- A rule can be considered a special case of a fact.
- A rule can be considered a special case of a goal.

1.4
- call-by-value
- call-by-reference
- return value
- all of them

1.5
- Use multiple return statements in the predicate.
- Use a single return statement to return a list that contains the multiple values required.
- Use multiple named variables to hold the values.
- Use multiple unnamed variables to hold the values.

1.6
- their predicates are the same.
- their arities are the same.
- their corresponding arguments match.
- all of the above are true.
1.7
- the head of the predicate.
- the body of the predicate.
- the neck of the predicate.
- the number of arguments of the predicate.

1.8
- a single rule.
- a single clause in a rule.
- the fact/rule base.
- the body of the next rule.

1.9
- is an imperative feature that should be discouraged.
- will cause a dead-loop for every goal.
- will cause a dead-loop when no match can be found.
- will never cause a dead-loop.

1.10
An anonymous variable in Prolog is a
- constant.
- placeholder.
- predicate.
- question.

1.11
- true?
- X = apple
- [apple, pear]
- None of them

1.12
?- child_of(mary, [amy | T]).
- T = [amy, david, conrad]
- T = david, conrad
- T = [david, conrad]
- T = conrad

1.13
- H = [amy]
- H = [david]
- H = david
- H = [amy, conrad]

1.14
- X = [81, 25, 9, 29]
- X = 9  Y = 29
- X = 9  Y = 81
- X = 81  Y = 400

1.15
The rules find
- the last element of a list.
- whether an element is a member of a list.
1.16
☐ all existing the backtrack points will be removed.
☐ all existing recursive exit points will be removed.
☐ all existing backtrack points and recursive exit points will be removed.
☐ none of the existing backtrack points and recursive exit points will be removed.

1.17
☐ Stop searching immediately.
☐ Remove all existing backtracking points.
☐ Stop search after the first match is found.
☐ Remove all backtracking and recursive exit points.

1.18
☐ Jump to the previous repeat predicate.
☐ Remove one backtracking point.
☐ Return false.
☐ Stop searching immediately.

1.19
☐ fails immediately.
☐ succeeds and adds a backtracking point.
☐ fails at the end of the database.
☐ fails when it is visited (executed) for the second time.

1.20
☐ It takes a single character from the keyboard and prints it.
☐ It repeatedly takes a character from the keyboard and prints it.
☐ It takes a character from the keyboard, prints it, and exits if the character is a digit.
☐ It takes a character from the keyboard, prints it, and exits if the character is NOT printable.

1.21
☐ Semantic Web performs semantic check of pages submitted to Web.
☐ Semantic Web enables keyword-based search.
☐ Semantic Web better supports automatic processing and integration of Web information.
☐ Semantic Web is based on Prolog.

1.22
☐ Prolog is frequently used to describe the semantic Web as a markup language.
☐ Prolog is frequently used for writing the parsers of the semantic description language.
☐ Prolog is a service-oriented programming language.
☐ All Prolog programs are semantic Web.
### 6.7 Homework, programming exercises, and projects

1. Multiple Choice. Choose only one answer in each question. Choose the best answer if more than one answer are acceptable.

1.1
- ☐ imperative
- ☐ object-oriented
- ☐ functional
- ☐ service-oriented

1.2 What language supports automatic garbage collection?
- ☐ C
- ☐ C++
- ☐ C#
- ☐ None of them

1.3
- ☐ service providers
- ☐ service brokers
- ☐ application builders
- ☐ None of them

1.4
- ☐ service providers
- ☐ service brokers
- ☐ application builders
- ☐ None of them

1.5
- ☐ preprocessor directives
- ☐ macros
- ☐ overriding
- ☐ switch statement

1.6
- ☐ multiple inheritance
- ☐ array
- ☐ pointer
- ☐ foreach statement

1.7
- ☐ declaration in C++
- ☐ printf statement
- ☐ pointer
- ☐ header files

1.8
- ☐ C
- ☐ C++
- ☐ C#
- ☐ Scheme

1.9
- ☐ control flow of Web services
- ☐ interface of Web services
- ☐ syntax of Web services
- ☐ semantics of Web services

1.10
- ☐ describing the interface of Web service
- ☐ composing SOA applications
- ☐ publishing Web services
- ☐ calling the remote Web services
1.1
- describing contact information of service provider
- describing the service types of the Web services
- describing technical details for remote invocation of Web services
- describing testing results on the reliability and trustworthiness of Web services

1.12
- describing the interface of Web service
- composing SOA applications
- publishing Web services
- calling the remote Web services

1.13
- A Web service is intended for being accessed by a computer program.
- A Web service is intended for being accessed by a human user.
- A Web application is intended for being accessed by a computer program.
- Web application is a synonym of Web service.

1.14
- It is identical to SOA software, and there is no difference.
- SaaS does not use SOA technology at all.
- SaaS is similar to SOA software; however, it is often hosted on a cloud environment.
- SaaS is the same as a Web service.

1.15
- software components and services.
- Application development environment.
- Computing capacity.
- Memory and disk space.

1.16
- Structured data
- Semi-structured data
- Unstructured data
- All of these types

1.17
- Noise elimination and fault tolerance
- Poly-structured data
- Real-time data processing
- Volatility of data

1.18
- Infrastructure
- Management.
- Analytic techniques
- Data types