SUNY Institute of Technology

Computer Science 431 Dr. R. Sarner
Spring 2023

Principles of Programming Languages

Required Text:

Mullick and Sarner, *Structure of Programming Languages*,
to be provided electronically without cost.

Course Objectives:

1. To become familiar with language evaluation criteria.

2. To understand language semantics, particularly operational semantics.

3. To understand module interfacing, parameter binding, primitives, data structures and control structures and recursion.

4. To understand translation and binding techniques and their associated advantages and disadvantages.

5. To understand the imperative, functional, and object-oriented, and logic language paradigms.

6. To understand the role of formal language definition and grammar.

7. To understand the semantic differences between selected high-level programming languages.

8. To be able to write significant programs in C or C++ and modest programs in two to four other selected high level programming languages.

9. To be able to read and understand code written in a variety of high level programming languages.

Course Structure:

This course is a combination of lecture, discussion, and outside work. Students are encouraged to raise questions at any time. The basic course material is covered in the text; additional material will be introduced via handouts, and will be posted on the Blackboard course site. Video recordings will be made of course sessions, and barring technical difficulties the video files will be available for viewing within 24 hours. However, video quality is not professional and recordings are not a substitute for class attendance. At the college level students should spend about two hours outside the classroom working on a course (a combination of reading, studying, and homework) for each hour of class meeting time. This means that students should expect to spend about eight hours per week outside the classroom working on CS 431. The course schedule (including topics, reading assignments, graded homework problems (GHPs) and exams) is listed at the end of this syllabus.

Office Hours:

I am committed to be available for students on as-needed basis. In addition to posted office hours (M/T/W 4:00-5:00 PM), I am generally available for consultation via Zoom on an appointment basis between 10:00 AM and 10:00 PM seven days a week. Zoom sessions are particularly effective for assisting students in debugging programs because using Zoom it is possible for me to see how the code acts on the student's own equipment. To schedule a Zoom session send me an email requesting an online appointment.

Accommodations for Students with Disabilities:

In compliance with the Americans with Disabilities Act of 1990 and Section 504 of the Rehabilitation Act, SUNY Polytechnic Institute is committed to ensuring comprehensive educational access and accommodations for all registered students seeking access to meet course requirements and fully participate in programs and activities. Students with documented disabilities or medical conditions are encouraged to request these services by registering with the Office of Disability Services. For information related to these services or to schedule an appointment, please contact the Office of Disability Services using the information provided below.

Office of Disability Services
(315) 792-7170

Utica Campus

Peter J. Cayan Library, L145

Students with a certified disability requiring extra time or an isolated setting must make arrangements with the Learning Center in advance of each exam. **The Learning Center will not accommodate students who have not made advance arrangements.**

Social and Professional Media:

I do not use or participate in social or professional media primarily because I find their privacy policies completely unacceptable. Consequently, I do not respond to requests to join networks such as LinkedIn because the simple act of responding gives the operator information that I do not believe they should have, and I do not use Facebook (personally or professionally). If I am sent a request to join your professional network do not take my refusal to respond as an indication of any disrespect toward you – it has nothing to do with you and everything to do with the terms of the service provider.

Letters of Reference and Reference Checks:

The Family Educational Rights and Privacy Act (FERPA) does not permit me to release any information regarding you or your performance in this course without your express consent. Consequently, I will not respond to requests from third parties (including your parents, guardians, or security checks by government agencies) for information about you unless they are in possession of a release signed by you, or you have contacted me directly and authorized me to proceed.

Plagiarism Warning:

The work you submit must be your own. You will not receive credit for work that is not your own. You can ask others for advice or help, but the actual coding, keyboarding, and running of your programs must represent your own work. **All sources of ideas that are used in any way (quoted, paraphrased, summarized, or adapted), including ideas taken from the text, must be acknowledged in program documentation. Failure to do so constitutes academic dishonesty and will result in a failing course grade**. **Submitting programs developed by others without your own modification or adaptation is expressly prohibited and will result in a failing course grade - whether or not it is attributed.** Obviously, the use of standard programming libraries such as stdio.h or iostream is permissible as is the use of the standard template library. Just like speeding on the Thruway, not all speeders are caught, and not all academic dishonesty is caught, but selective enforcement does not constitute a valid defense.

Grades:

Grades will be based on the following items:

Eight programming assignments; six worth 30 points each, and two worth 60 points each for a maximum total of 300 points. The completion of **eight** assignments is a necessary but not sufficient condition for receipt of a passing course grade. Four of the six 30 point assignments will be chosen from short assignments in the languages covered in the second half of the course. Specific criteria for grading assignments appear later in this syllabus.

Three in-class exams will be given. Each exam is worth 150 points. The two highest exam scores will be counted. Exams will be announced at least one week in advance; no exams will be given during the last class week.

A comprehensive final exam worth a maximum of 200 points will be given during the regularly scheduled final exam period.

A group project consisting of both an in-class presentation (target length 60-90 minutes) and the submission of a paper with a maximum value of 200 points, 100 points for the presentation and 100 points for the paper..

Students may have an opportunity to earn a maximum of an additional 50 points by creating potential final exam questions.

Points from the exams, assignments, project, and extra credit will be aggregated and grades will be assigned employing the following scale:

Points	Grade
950+	A
850-949	B
750-849	C
650-749	D

No "+" or "-" grades will be given. Incomplete grades may be given at the discretion of the instructor. Students taking an "I" grade should be aware:

1. The student must personally meet with the instructor and have the remaining programs graded in the student's presence. The student has the responsibility for securing the required appointment.

2. "I" grades that are not resolved automatically convert to "F" grades at the middle of the Fall 2023 semester. Students requiring a further extension must secure it in writing.

Topical Outline:

I. Language Evaluation

II. Operational Semantics
A. Flow chart language
B. Environments
C. Data Sharing
E. Storage Allocation
F. Primitives
G. Control Structures
H. Recursion
I. Data Structures

III. Language Paradigms
A. Imperative Paradigm
B. Functional Paradigm
C. Object-Oriented Paradigm
D. Logic Programming

IV. Formal Definition and Grammar

V. Language Survey (illustrative, based on number of groups)
A. FORTRAN
B. APL
C. Prolog
D. Haskell
E. LISP
F. J
G. Python
H. COBOL
I. Ruby
J. Oberon
K. Delphi
L. Rebol
M. PHP
N. Scala
O. IO
P. Erlang
Q. Clojure
R. Lua
S. Elm
T. Julia
U. Idris
V. Elixir
W. Factor
X. MiniKanren
Y. R
Z. Rust

Standards for Assignments:

Programs must be submitted as attachments through Angel. Source code for all assignments must be submitted. Assignments must compile and execute in the environment designated for that assignment.

Programs will be graded with a maximum point value as indicated. Students are expected to produce professional quality programs adhering to the following criteria:

(1) The problem must be completely solved. The sophistication of the solution will be considered in determining the grade. All paths through the program must produce correct results. **A solution that minimally solves the problem gets a minimal grade, and hence creative enhancements are encouraged.** Keep in mind, though, that an enhancement that does not work properly results in a significant penalty.

(2) The documentation must be complete and the program layout must be visually appealing. **Each program and each subprogram (procedure or function) must contain a statement of purpose, name of author, language and dialect, list of identifiers, dependent subunits, date of creation, revision number (if any), date of last revision, and citation of sources**. Intra-code commenting for unusual code is expected. Identifier names must be rational. **The use of correct grammar and spelling in user prompts is assumed; the penalty for sloppy English in the user interface will be harsh.**

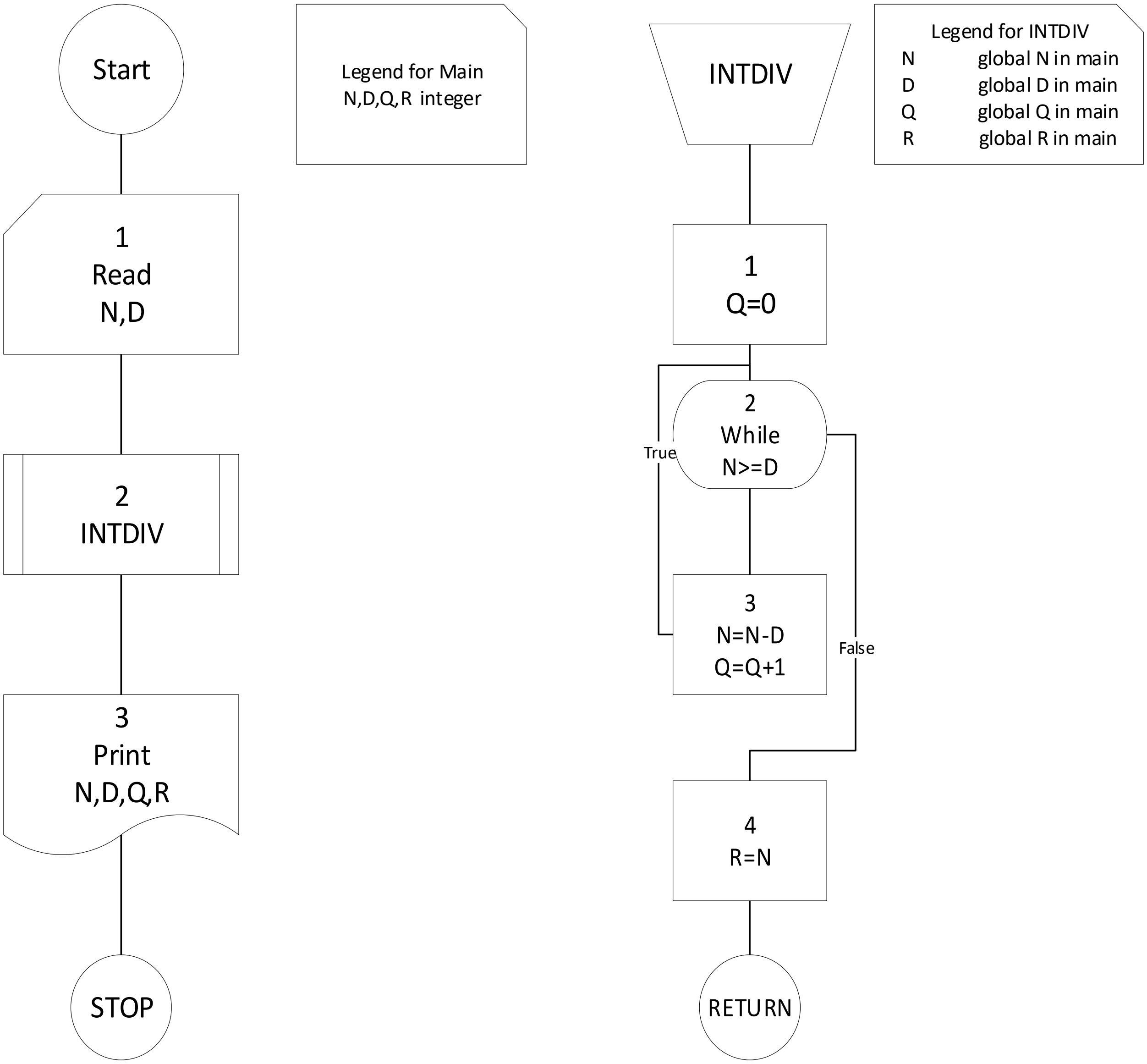
(3) Assignments 1-3 must be crash-proof. User prompts must be clear, precise, grammatically correct and spelled correctly. In the absence of warnings any user input is considered fair game. You should not expect the dumb user (me) to remember a complex set of instructions; programs should be *user-friendly*.

(4) Programs must be submitted on-time. There will be a penalty of 20% of the maximum point value for programs that are submitted late. Programs will not be graded prior to the due date.

Programming Assignments:

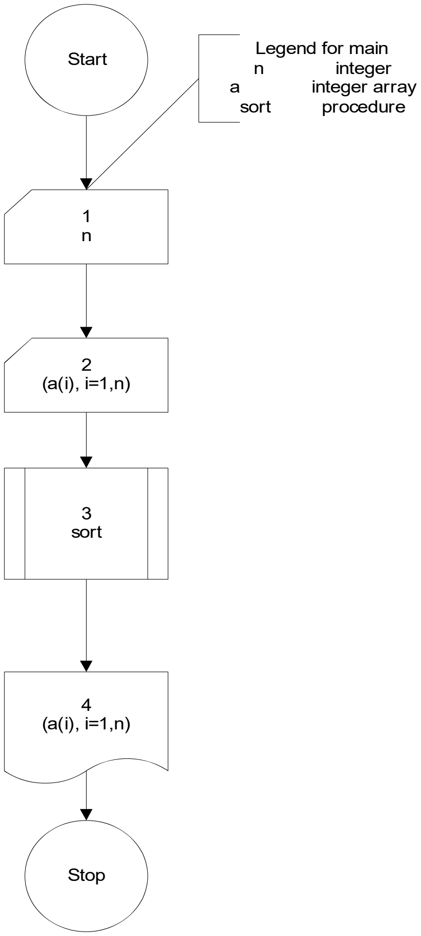
Assignment #1 (due 2/10) - 30 points

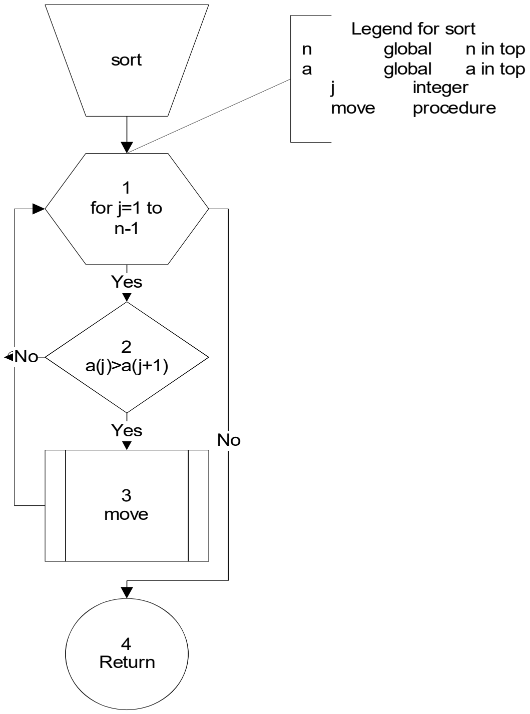
The process of integer division involves two positive integers, N and D, finding a non-negative result Q represents the quotient and R represents the remainder. Write a C or C++ program that successfully compiles and executes using gcc or g++ using the following flow chart. You **must** scope the variables as indicated (even though global variables rot) as indicated, and **must** get the values for N and D from the user.
The point of this program is to convince you - assuming you really need convincing - that global variables are a lousy practice.



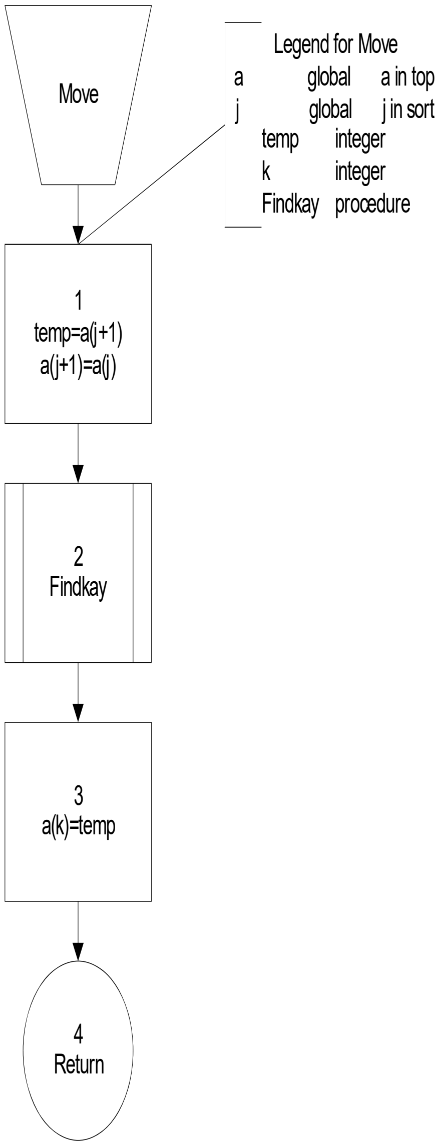
Assignment #2 (due 2/24) - 30 points

Write a C or C++ program that operationalizes the (awful) sort shown in the following flow chart. You **must** use this sorting algorithm, with variables scoped as indicated in the flow chart. Your program must compile and execute under the UNIX gcc or g++ compiler.



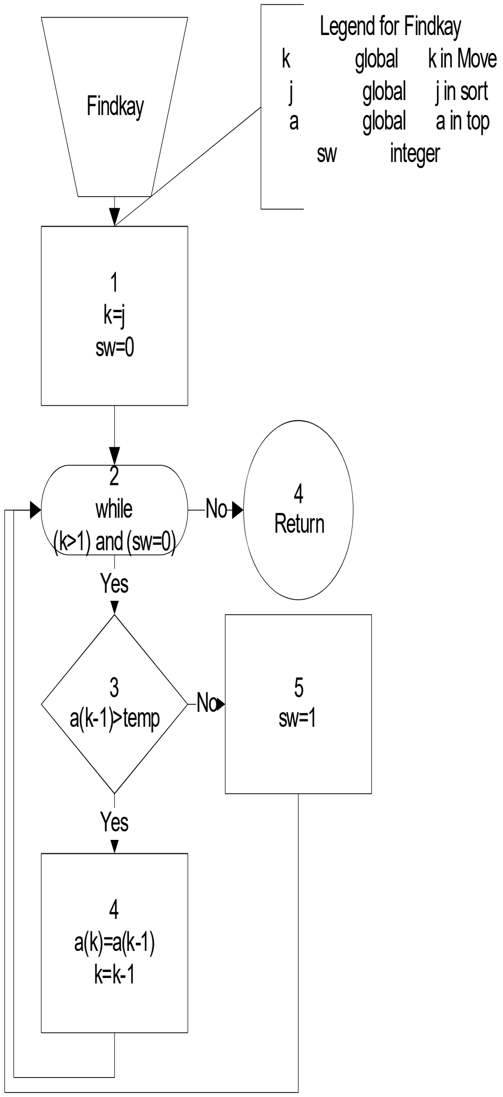


FLOWCHART CONTINUED ON NEXT PAGE



Legend for Move

a	global	a in top
j	global	j in sort
temp	integer	
k	integer	
Findkay	procedure	



Legend for Findkay

k	global	k in Move
j	global	j in sort
a	global	a in top
sw	integer	

Assignment #3 (due 3/31) - 60 points

Write a C or C++ program **incorporating recursion** to solve the following problem. Your program must compile and execute under the UNIX gcc or g++ environments.

Accept a sequence of k characters from the user and use the characters to produce a k concentric squares. Assuming an index origin of 1, the outermost square is made up of the letter a[k], the square inside that is made from the letter a[k-1], the square inside that is made from the letter a[k-2], and so on, with the innermost square made from the letter a[1]. For example, if k=5, and the values of a are
("R","O","C","K","Y") then the executed program should display:

```
Y Y Y Y Y Y Y Y Y
Y K K K K K K K Y
Y K C C C C C K Y
Y K C O O O C K Y
Y K C O R O C K Y
Y K C O O O C K Y
Y K C C C C C K Y
Y K K K K K K K Y
Y Y Y Y Y Y Y Y Y
```

Hint: This will be simplest to do if you implement it in C++ and write your own class; instances of this class can be passed by value. If you do not use classes you will have to find an alternative to passing an array by reference.

Assignment #4 (due 4/14) - 60 points

Write a program in an object-oriented language (either C++ or FORTRAN95):

1. Queries the user for the name of a file of text.
2. Opens the file, and maintains two lists: one list for words beginning with the letter "D" or "d", and a second list for words beginning with any other letter. Each list must maintain words in alphabetical order.
3. Each node in the list must contain the word and the number of times that the word appears.
4. Display (a screen at a time) each of the lists showing the alphabetized list of words and the number of times that each appears.

This assignment will be tested using the file **dtext.txt** available on Blackboard.

Assignments #5 through Assignment #8

Select the assignments distributed with any four of the languages covered and submit your solution to these problems. Due dates will be announced.

Group Project

Each group will be assigned a language. The group will be required to produce both a paper and a presentation on their assigned language. A typical presentation will be a full class duration. The paper is to be submitted as a pdf document and will be shared with all class members through Angel. A typical outline of the language:

I. History
II. Distinguishing Features and Application Domain
III. Data Types
IV. Primitives
V. Scoping and Parameter Passing
VI. I/O Functionality
VII. Control Structures including Recursion
VIII. Data Structures (including arrays, ADTs)
IX. Assessment of language employing Tucker's Criteria
X. Sample Programs
XI. Live demonstration of the language environment.

Individual grades on the group project will be determined by weighting the group grade by a factor determined by aggregating evaluations of the contributions of each group member submitted by other members of the group.

Each group will also have to define a programming assignment in that language for the rest of the class. The assignment should be something that a typical student could be expected to complete in not more than a couple of hours excluding any required software installation. The group will be responsible for grading all submissions of that assignment and reporting the results to the instructor.