



MAJOR FIELD TESTS

Colleges and universities use the Major Field Tests to measure student academic achievement and growth and to assess the educational outcomes of their major programs. In addition, academic departments use the Major Field Tests to evaluate their curricula and to measure the progress of their students. The tests also provide students with an assessment of their own level of achievement within a field of study compared to that of students in their program and to national comparative data.

Background

Development of the Major Field Tests began in 1989, modeled on the development of the Graduate Record Examinations® (GRE®) Subject Tests. However, unlike the GRE Subject Tests, the Major Field Tests do not serve as a predictor of graduate school success, but are designed to measure the basic knowledge and understanding achieved by senior undergraduates in their major field of study. Each test is revised approximately every five years. Experienced teaching faculty members representing all the relevant areas of a discipline participate in determining test specifications, questions, and types of scores reported. ETS assessment experts subject each question to rigorous tests of sensitivity and reliability. In addition, every effort is made to include questions that assess the most common and most important topics and skills within each major field of study.

Test Content

The Major Field Tests are designed to assess mastery of concepts, principles, and knowledge expected of students at the conclusion of an academic major in specific subject areas. In addition to factual knowledge, the tests evaluate students' abilities to analyze and solve problems, understand relationships, and interpret material. The tests may contain questions that require interpretation of graphs, diagrams, and charts based on material related to the field. Academic departments may add up to 50 additional locally written questions to test areas of a discipline that may be unique to the department or institution.

Test Length

All Major Field Tests are multiple-choice exams lasting two hours (three hours for MBA), and administered in a proctored environment. However, the addition of optional locally developed questions may result in a longer testing period.

Test Administration

Departments or schools choose when and where to give the tests; however, the tests are normally administered during the senior year when students have completed the majority of courses in the major. Many institutions administer the tests as part of the requirements of a capstone course.

National Comparative Data

A Comparative Data Guide, published each year, contains tables of scale scores and percentiles for individual student scores, departmental mean scores, and any subscores or group assessment indicators that the tests may support. The tables of data are drawn from senior-level test takers at a large number of diverse institutions. More than 500 colleges and universities employ one or more of the Major Field Tests for student achievement and curriculum evaluation each year.

Scores

Major Field Test score reports are sent directly to the office within an institution that purchases them, such as a department chairperson, dean, or director of testing. Results of the tests are reported for the entire group of test takers, as well as for individual students. Overall student scores are reported on a scale of 120–200; subscores (which many of the tests include) are reported on a scale of 20–100. Another score reported for most of the tests is based on group-level achievement in subfields of the discipline. These “assessment indicators” report the average percent of a subset of test questions answered correctly by all students tested. On Major Field Tests, only correct answers are scored, so students are not penalized for omissions or guesses.

COMPUTER SCIENCE (4CMF)

(Current form introduced in January 2006)

The Major Field Test in Computer Science consists of 66 multiple choice questions, some of which are grouped in sets and based on such materials as diagrams, graphs, and program fragments. The outline below shows the content areas covered on the test and the approximate distribution of questions among content areas.

- I. **Discrete structures** (15–21 percent)
 - A. Functions, relations, and sets
 - B. Basic logic
 - C. Proof techniques
 - D. Basics of counting and number theory
 - E. Graphs and trees
 - F. Discrete probability
- II. **Programming** (21–27 percent)
 - A. Programming fundamentals
 1. Fundamental programming constructs
 2. Basic algorithms and problem solving
 3. Fundamental data structures
 4. Recursion
 5. Event-driven programming
 6. Object-oriented programming
 - B. Programming languages
 1. Features, paradigms, implementation techniques
- III. **Algorithms and complexity** (16–22 percent)
 - A. Advanced data structures and algorithms (including graph algorithms)
 - B. Algorithmic strategies
 - C. Distributed algorithms
 - D. Basic computability and complexity
 - E. Automata theory
- IV. **Systems** (16–24 percent)
 - A. Architecture
 1. Digital logic and digital systems
 2. Machine level representation of data
 3. Assembly level machine organization
 4. Interfacing and communication
 - B. Operating systems
 1. Operating system principles
 2. Concurrency
 3. Scheduling and dispatch
 4. Memory management
 - C. Networking

- V. **Software engineering** (3–9 percent)
 - A. Software requirements, specifications, design, validation, and management
- VI. **Information management** (3–8 percent)
 - A. Database systems
 - B. Data modeling
- VII. **Other** (3–8 percent)
 - A. Human computer interaction
 - B. Graphics
 - C. Intelligent systems
 - D. Social and professional issues
 - E. Web computing

MFT Computer Science Pseudocode Statement

Scores on the Computer Science Test are reported as follows:

Total Score

Reported for each student and summarized for the group.

Assessment Indicators

Reported for the group* only.

- Programming (22)
- Discrete Structures and Algorithms (25)
- Systems (Architecture, Operating Systems, Networking, Database) (16)

Numbers in parentheses are approximate number of questions in each category.

*A minimum of five students is required for Assessment Indicators to be reported.



Copyright © 2006 by Educational Testing Service. All rights reserved. EDUCATIONAL TESTING SERVICE, ETS, the ETS logo, GRADUATE RECORD EXAMINATIONS, and GRE are registered trademarks of Educational Testing Service. HIGHER EDUCATION ASSESSMENT is a trademark of Educational Testing Service.

Permission to reproduce this document is hereby granted to institutions (colleges and universities) administering the Major Field Tests for internal use only. No commercial or further distribution is permitted. Other persons or agencies wishing to obtain permission to reproduce this material may write to the Permissions Administrator at Educational Testing Service, Princeton, New Jersey 08541.

MFT Computer Science Pseudocode Statement

We currently do not use any specific programming languages in questions on the MFT Computer Science Exam. Instead, we use a simple pseudocode that we believe will be easily understood by any computer science student. See the examples below.

Example 1. Class declaration and object instantiation

```
class StudentInfo
    int studentID
    string name
end class StudentInfo

StudentInfo x ← new StudentInfo()
x.studentID ← 1234 //the value 1234 is assigned to x.studentID
x.name ← "John"
print ( x.studentID )
print ( x.name )
```

Example 2. The following procedure swaps the values of two parameters.

```
swap ( pass-by-reference int x, pass-by-reference int y )
    int temp ← x
    x ← y
    y ← temp
end swap
```

Example 3. SelectionSort

Preconditions: A is an array of integers.

The length of array A is n.

The index of array A starts at 0.

```
int[] selectionSort ( pass-by-reference int[] A, int n )
    int min
    int j
    int i ← 0
    while ( i ≤ n - 1 )
        min ← i
        j ← i + 1
        while ( j ≤ n - 1 )
            if ( A[j] < A[min] )
                min ← j
            end if
            j ← j + 1
        end while
        if ( min ≠ i )
            swap ( A[min], A[i] )
        end if
        i ← i + 1
    end while
    return A //returns the sorted array
end selectionSort
```