

INITIAL DESIGNER TRAINING (IDT)

Evaluation Testing

Student Handbook

Revised November 1989



## Table of Contents

Item	Page
Introduction.....	1
Purpose of IDT Testing.....	2
Job Related Tests.....	2
Test Requirements.....	3
Test Objectives.....	4
Test Schedule.....	5
Test Givens.....	7
Code of Conduct.....	8
Your Responsibilities.....	8
Work Hours.....	9
Role of the Instructor.....	10
Performance Evaluation.....	11
Walkthroughs.....	12
Review Limitations.....	14
Time Tracking.....	14
Test Pass/Fail Criteria.....	15
Performance Feedback.....	16
IDT Pass/Fail.....	16
Appendices	17
1 - Checkpoint Products "C" CODE CHECKLIST	23
2 - Test Product Checklists	

---

Introduction

The intent of this handbook is to provide you with important information about the tests you will be taking as you progress through IDT. We want you to be aware of the objectives of the tests, how the tests are conducted, and the ground rules that apply.

Before starting each test it is suggested that you carefully review the contents of this package. Any remaining or additional questions should be addressed to your instructor.

---

Purpose  
of IDT  
Testing

---

Testing in the context of IDT serves these important functions:

- provide the opportunity for students to exercise and reinforce skills taught in the course
- acquaint students with job requirements and expectations
- confirm that students have adequately learned critical performance skills and can satisfy fundamental on-the-job requirements

The last item represents the pass/fail nature of the course. Students unable to demonstrate satisfactory performance during training lack the skills needed to meet the more rigorous on-the-job expectations. Failure to satisfactorily complete IDT means the student will not retain employment as an Application Developer.

---

Job Related  
Tests

The tests used in IDT were carefully designed to be consistent with course material and to reflect actual job activities performed by an Application Developer.

Each test requires the use of a wide range of skills needed to accomplish the major Application Developer's job duties. Specifically, the job duties performed by an Application Developer include:

1. design data processing solutions
2. code data processing solutions
3. test data processing solutions
4. maintain existing code
5. document data processing solutions
6. manage time and resources

---

Continued on next page

---

Job Related  
Tests (cont.)

The goal of testing is to simulate on-the-job conditions and requirements. The most effective means of achieving this objective requires tests that are performance oriented. The format of IDT tests is one that provides a problem statement or "job-specs" describing a business need which can be met by a computer solution. Your task is to develop a solution using the performance skills taught in the course.

---

Test  
Requirements

On completion of a test, your solution to the defined problem must satisfy specific criteria.

In general, these are the critical requirements:

- the solution produces correct output
  - the solution is complete - all required products are documented and submitted for review
  - the solution is completed on or before schedule
  - required products reflect an acceptable level of quality
-

---

Test  
Objectives

While a detailed breakdown of evaluation criteria appears in a later section, the following list outlines general objectives for each of the six Application Developer's job duties:

<u>Job Duty</u>	<u>Objective</u>
design	produce clear, workable, structured design using AT&T guidelines
code	produce workable, structured code that adheres to AT&T guidelines
test	thoroughly identify necessary test conditions and verify accuracy of program results
maintain	correct and/or modify existing code as required
document	satisfactorily document solutions according to AT&T guidelines
manage time and resources	successfully complete assignment on or before schedule.

---

Test  
Schedule

---

The schedule of tests in IDT coincides with the outline of the course - which consists of three phases of learning and skills development. At the conclusion of each phase a major test is administered in order to measure the degree of performance ability.

Highlights of each test:

Test 1

This test occurs relatively early in the course and measures basic design and programming skills. It is administered in two parts. Part I is assigned after the design units and represents the design component of the test; Part II involves coding and testing a program based on the design produced in Part I.

Test 2

Test 2 is in many respects similar to Test 1 except that it measures more sophisticated design and programming skills. The job specs provided require multi-module programming.

NOTE: Tests 1 and 2 measure a student's intermediate progress and are used to provide feedback. Students failing to satisfy critical objectives will be counseled that their performance is unacceptable.

---

Continued on next page

Test  
Schedule  
(cont.)

Case Problem

Tests 3 and 4 together comprise the end of course case problem.

Test 3 represents a debugging/maintenance problem in which you are given a coded program that contains errors. You must repair the code and then apply a maintenance change according to updated job spec requirements.

Test 4 involves a multi-module, multi-jobstep problem that demands the use of table handling concepts and skills. The format follows that of Tests 1 and 2; high level job specs are provided from which you must prepare a design solution and then code and test the resulting programs.

NOTE: The case problem covers the critical job activities performed by Application Developers and, as such, it provides the basis for determining a student's pass/fail status for the course.

---

Test  
Givens

---

To enable you to satisfy test requirements you are provided access to the tools and materials typically used in arriving at a computer solution.

The resources at your disposal include:

- Terminal access - use of Terminal during the tests is restricted to a pre-determined schedule which will be posted in the classroom
- relevant reference materials, such as:
  - Language coding manuals and system references
  - DSPs/SDPLs
  - "The Practical Guide to Structured Systems Design"
  - IDT course materials
  - your own class notes
- job aid materials, e.g.:
  - coding sheets
  - documentation forms
  - templates, etc.
- instructor clarification of test specs and requirements (see page 10 - role of the instructor)
- checkpoint reviews (see page 11 - performance evaluation)

You are expected to be familiar with all available resources and to use them effectively.

---

---

Code  
of  
Conduct

The resources identified above do not include assistance from other students; YOU ARE NOT FREE TO WORK IN GROUPS OR SHARE INFORMATION. The solution you submit on each test must be your own - to do otherwise represents a violation of the AT&T Code of Conduct. Any individual violating the code of conduct will be subject to discipline, up to and including dismissal.

---

Your  
Responsibilities

Successful completion of IDT rests with you. Each test clearly specifies the activities and products required. In addition, you are expected to:

- follow test directions closely; observe all test requirements
  - carefully read and analyze test specs before starting your solution
  - identify items in the job specs that are unclear to you and obtain clarification from your instructor (if there is an element of doubt, do not assume)
  - use available resources to the fullest
- 

Continued on next page

---

Your  
Responsibilities  
(cont.)

- plan and organize your work effectively; e.g. if terminal access is not available, then begin work on another task - such as preparing documentation or test plans
- produce complete test solutions; do not overlook details

---

Work  
Hours

The IDT course was designed to allow completion of exercises, assignments, and tests during normal working hours. You may, at your discretion, work beyond scheduled classroom time; however, discretionary work time will not be compensated.

---

---

Role  
of the  
Instructor

When a test is in progress your instructor assumes the role of "system manager" or "project leader" and as such is available to answer questions about job specs and system requirements. The instructor is not free to offer advice or information about how to solve the test problem. However, in the event you encounter a problem that escapes resolution after you have exhausted all available resources (e.g. unexplained OUTPUT), you can request the assistance of your instructor. At that point s/he will make an independent determination of the cause of the problem. If the problem is indeed one that falls outside the scope of the course (requires technical knowledge not covered in IDT), your instructor will provide direction that will enable you to reduce the problem to a point within the technical realm of the course; and from there you are again on your own.

Remember, it is your responsibility to use available resources and to make a concentrated effort to solve the problem before approaching the instructor.

---

Performance  
Evaluation

---

Your performance on a test is evaluated at specific checkpoints while the test is in progress. A test checkpoint simply represents a completed phase of program development at which time critical test products will be reviewed for completeness and quality.

There are typically four test checkpoints:

1. design walkthrough
2. test plan review
3. coding review
4. test results review

The products evaluated at each checkpoint are listed in Appendix 1. Each product is evaluated on the basis of completeness and adherence to AT&T guidelines, as defined in IDT course materials. Checklists outlining specific criteria for each test product appear in Appendix 2. Because successful test performance is contingent on an acceptable level of quality, please carefully review the checklists and make every attempt to develop your test products in accordance with the criteria listed. If checkpoint review uncovers unacceptable errors in one or more test products, you will be required to make needed corrections. Unless identified errors are corrected you will not be permitted to complete the test. A large number of errors may cause you to lose time and result in failure to meet the on-time requirement.

---

---

## Walkthroughs

Checkpoint reviews are often performed in the context of a walk-through.

In a walkthrough, the student and instructor have clearly defined roles.

### Student's Role

- prepare and make copies of products to be reviewed
- schedule an appointment for the walkthrough with your instructor
- provide a copy of all walkthrough products to your instructor in advance of the walkthrough
- verbally describe each product at the walkthrough; provide clear information about what it accomplishes and how
- respond to instructor questions
- record all identified problem areas or items that need modification
- sign the checkpoint status report
- if needed, make corrections and schedule another walkthrough

---

Continued on next page

---

Walkthroughs  
(cont.)

Instructor's Role

- review required products
- determine whether the reviewed products actually accomplish the purpose of the job specs; where errors appear to exist, question the student to obtain clarification or point out the apparent problem
- determine the status of the walkthrough at its conclusion, i.e. reviewed products are accepted or rejected.
- Sign the checkpoint status report

As previously noted, instructors are not free to indicate how to resolve detected errors - that is your responsibility.

---

---

Review  
Limitations

Although checkpoint reviews and walk-throughs are intended to help identify major errors, it is not the role of the instructor to uncover all possible errors. Walk-through acceptance is not a guarantee that your product(s) will work as expected. Dynamic testing helps reveal unexpected problems; it is your job to locate and resolve all problems in order to produce valid program results.

---

Time  
Tracking

Each test requires that you keep a record of the time, actual hours spent on specific products, by using the Test Timing form. The form is attached to each test as the last page. Refer to instructions on the Test Timing form for information on recording your time.

---

---

**Test  
Pass/Fail  
Criteria**

Your performance on each test is judged on the basis of four critical requirements:

1. Checkpoint products are accepted by the instructor, i.e. each product reflects an acceptable level of quality (this criterion does not apply on test 3).
2. The test solution is complete - all required products are documented and submitted for review.
3. The test solution produces correct output.
4. The test solution is completed on or before schedule.

All applicable criteria must be satisfied for the student to obtain a passing status for the test. Failure to satisfy one or more criteria automatically results in a failing status.

---

---

Performance  
Feedback

At the conclusion of each test and after your instructor has evaluated the test results, s/he will schedule an appointment with you to review your performance. Apparent strengths and weaknesses will be discussed; and where performance approaches an unacceptable level your instructor will provide some suggestions on what you can do to try to improve it.

---

IDT  
Pass/Fail

A student's pass/fail status for the course is determined at the conclusion of the case problem (tests 3 and 4). Failure on either test 3 or test 4 represents unacceptable performance and disqualifies the student for the Application Developer position.

---

## TEST PRODUCTS REQUIRING INSTRUCTOR ACCEPTANCE

---

Appendix 1

Checkpoint Step:      Products

Design

1. structure chart
  2. load module hierarchy
  3. implementation plan
  4. psuedocode
- 

Test Plans

5. test plans, including
    - summary of test cases
    - test data
    - expected results
- 

Code

6. compiled listing of appropriate programming language
- 

Test Results

7. unit/verification tests, including:
    - C-LISTS (MVS) or Shells (UNIX)
    - output results
-

## DESIGN LOGIC CHECKLIST

Appendix 2

Product	Items Reviewed	Criteria
Structure Chart	<ul style="list-style-type: none"> <li>• logical relationship between functions</li> <li>• symbols and format                             <ul style="list-style-type: none"> <li>- function boxes</li> <li>- data couples</li> </ul> </li> <li>• conventions                             <ul style="list-style-type: none"> <li>- function numbers</li> <li>- function names (single verb/object)</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• clear representation of functions (i.e. reasonable degree of factoring)</li> <li>• correct implementation of structure chart guidelines (e.g. reasonable degree of cohesion)</li> </ul>
Load Module	<ul style="list-style-type: none"> <li>• structure chart - LMH relationship</li> <li>• format</li> <li>• symbols</li> <li>• conventions</li> </ul>	<ul style="list-style-type: none"> <li>• matches structure chart functions (names and numbers)</li> <li>• correct implementation of LMH guidelines</li> <li>• identifies existing and new reusable modules</li> </ul>

## DESIGN LOGIC CHECKLIST

Appendix 2

Product	Items Reviewed	Criteria
Pseudocode	<ul style="list-style-type: none"><li>• structure chart - pseudocode relationship (function numbers, names, etc.)</li><li>• structured words format<ul style="list-style-type: none"><li>- single page per function</li><li>- indentation</li></ul></li><li>• conventions<ul style="list-style-type: none"><li>- embedded functions</li><li>- function invocation</li></ul></li></ul>	<ul style="list-style-type: none"><li>• matches structure chart</li><li>• correct implementation of pseudocode guidelines</li></ul>
Implementation	<ul style="list-style-type: none"><li>• list of existing modules and new modules</li></ul>	<ul style="list-style-type: none"><li>• plan is clear and understandable</li></ul>

## TEST PLAN CHECKLIST

Appendix 2

The following list applies to each type of test plan:  
module, integration, job stream.

Product	Items Reviewed	Criteria
Summary of Test Cases	<ul style="list-style-type: none"><li>• black box test cases, where applicable</li><li>• grey box test cases</li><li>• white box test cases - added to test plan after code is written and compiled</li></ul>	<ul style="list-style-type: none"><li>• summary is complete, i.e. it matches pseudo-code conditional processing and addresses black box, white box, and boundary conditions where applicable</li></ul>
Test Data	<ul style="list-style-type: none"><li>• test data and test run descriptions</li></ul>	<ul style="list-style-type: none"><li>• matches identified test cases</li><li>• groups data into test runs</li><li>• includes a brief abstract of key conditions tested by each run</li></ul>
Expected Results	<ul style="list-style-type: none"><li>• test data - expected results relationship</li></ul>	<ul style="list-style-type: none"><li>• matches identified test runs</li></ul>

Note Due to the amount of information and the level of detail involved in a test plan, the review process focuses only on high level content; i.e. all required products are accounted for and demonstrate face validity. Ultimate responsibility for completeness and accuracy rests with the student.

# COBOL CODE CHECKLIST (MVS)

Appendix 2

Product	Items Reviewed	Criteria
ID/Environment Divisions	<ul style="list-style-type: none"> <li>• Program - ID</li> <li>• Date Written</li> <li>• Proprietary Markings</li> </ul>	<ul style="list-style-type: none"> <li>• adheres to standards as outlined in DSP 26.1-4</li> </ul>
Data Division	<ul style="list-style-type: none"> <li>• BLOCK CONTAINS clause</li> <li>• program version ID</li> <li>• data naming conventions</li> </ul>	<ul style="list-style-type: none"> <li>• adheres to standards as outlined in DSP 26.1-4</li> </ul>
Procedure Division	<ul style="list-style-type: none"> <li>• pseudocode - procedural code relationship</li> <li>• paragraph names use of comments</li> <li>• source code indentation</li> <li>• use of literals</li> <li>• use of switches</li> <li>• source code spacing</li> <li>• implementation of permissible control structures</li> </ul>	<ul style="list-style-type: none"> <li>• matches design logic</li> <li>• adheres to standards as outlined in DSP 26.1-4</li> </ul>

## COBOL CODE CHECKLIST (MVS) (Continued)

Appendix 2

Product	Items Reviewed	Criteria
Included Documentation	<ul style="list-style-type: none"> <li>all relevant sections</li> </ul>	<ul style="list-style-type: none"> <li>complete and correct</li> </ul>
Test Plan Addendum - White Box Test Cases	<ul style="list-style-type: none"> <li>list of test cases, test data, and expected results</li> </ul>	<ul style="list-style-type: none"> <li>test cases match COBOL condition statements that are derived as a result of coding implementation.</li> </ul>

## "C" CODE CHECKLIST (UNIX)

Appendix 2

Product	Items Reviewed	Criteria
Header Files	<ul style="list-style-type: none"><li>• token replacements (# define)</li><li>• structure declaratives</li></ul>	<ul style="list-style-type: none"><li>• adheres to standards</li></ul>
Main Function and Sub Functions	<ul style="list-style-type: none"><li>• file inclusions (# include)</li><li>• required documentation name title what how entry exit</li><li>• function names</li><li>• use of comments</li><li>• source code indentation</li><li>• use of pointers</li><li>• use of string handling functions</li><li>• use of standard library routines</li></ul>	<ul style="list-style-type: none"><li>• adheres to standards</li><li>• matches design logic</li></ul>

## "C" CODE CHECKLIST (UNIX) (Continued)

Appendix 2

Product	Items Reviewed	Criteria
Makefiles	<ul style="list-style-type: none"><li>• macro definitions</li><li>• dependancy lines</li><li>• executable commands</li><li>• comments</li></ul>	<ul style="list-style-type: none"><li>• adheres to standards</li></ul>
White Box Test Cases	<ul style="list-style-type: none"><li>• list of test cases, test data, and expected results</li></ul>	<ul style="list-style-type: none"><li>• test cases match conditional statements</li></ul>

# UNIT/VERIFICATION TEST RESULTS CHECKLIST

Appendix 2

Product	Items Reviewed	Criteria
CLIST's (MVS) or Shells (UNIX)	<ul style="list-style-type: none"><li>• command syntax</li><li>• conditional processing</li></ul>	<ul style="list-style-type: none"><li>• workable</li></ul>
Output results	<ul style="list-style-type: none"><li>• module test runs</li><li>• integration and job test runs, when applicable</li></ul>	<ul style="list-style-type: none"><li>• matches test plan correct</li></ul>