ABSTRACT

This project consists of a program to create and maintain a test bank, a program to create tests, and a test bank containing questions associated with the high school course Computer Mathematics I.

The programs are designed to be teacher utilities, and may be used to create tests related to the essential elements of any subject (English, math, etc).

The Texas Education Agency has defined 7 essential elements with 51 sub-elements as being the minimum topics taught in a Computer Mathematics I course in the state of Texas. The agency also requires that teachers be able to document the essential elements mastered by a student.

The programs in this project have been designed to help the teacher test and document essential element mastery. The test bank program requires that all questions pertaining to one sub-element be entered together and saved to a file named by the sub-element number. The test generator program prints a cover sheet containing a list of the sub-elements tested, labels each group of questions by sub-element, and labels the answers by sub-element.

The test bank program allows 5 types of questions: multiple choice, true/false, valid/invalid, short answer, and discussion/demonstration.

The test generator program allows the teacher to make tests that contain one or more essential elements and one or more question types.

The test bank disk contains files of questions pertaining to the essential elements of Computer Mathematics I.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Project Development</td>
<td>3</td>
</tr>
<tr>
<td>Environment</td>
<td>5</td>
</tr>
<tr>
<td>Logical Design</td>
<td>6</td>
</tr>
<tr>
<td>Module Descriptions</td>
<td>24</td>
</tr>
<tr>
<td>File Descriptions</td>
<td>56</td>
</tr>
<tr>
<td>Summary</td>
<td>59</td>
</tr>
<tr>
<td>Bibliography</td>
<td>62</td>
</tr>
<tr>
<td>Appendices</td>
<td>65</td>
</tr>
<tr>
<td><strong>A: Computer Mathematics &amp; Essential Elements</strong></td>
<td>66</td>
</tr>
<tr>
<td><strong>B: Question Formats</strong></td>
<td>68</td>
</tr>
<tr>
<td><strong>C: Program Documentation</strong></td>
<td>69</td>
</tr>
<tr>
<td><strong>D: Sample Tests</strong></td>
<td>90</td>
</tr>
<tr>
<td><strong>E: Program Listings</strong></td>
<td>117</td>
</tr>
<tr>
<td><strong>F: Test Bank Contents</strong></td>
<td>185</td>
</tr>
</tbody>
</table>
INTRODUCTION

The author of this project is a Computer Mathematics teacher who was seeking an easier method of creating exams. In the past, tests were generated and saved for later reference using a word processor. This method did not allow the desired versatility and flexibility.

The state of Texas, through the Texas Education Agency, established standards called essential elements for all subjects. House Bill 72 required that a student master 70% of these essential elements to receive a passing grade. The state, through the Texas Education Agency, required that teachers document student mastery of the essential elements. The author's superiors decided the best way to achieve this was to link test questions to the essential elements.

In searching for a solution to these problems - versatility, flexibility, and linking to essential elements - the author investigated many test banks furnished by textbook companies and many test generator programs. All of these had limitations.

Test banks furnished by book companies had questions organized by the chapters in the book they accompanied. The author was seeking questions organized by essential elements.

Test generator programs did not have the speed or versatility desired by the author.

The goals of this project are:

1. a fast, versatile test generator,
2. a test bank editor to allow for the creation and maintenance of test questions,
3. a test bank with questions for the course.
Computer Mathematics I, and
4. an overall link with essential elements in general,
and the essential elements of Computer Mathematics I
in particular.

In addition, the author hopes this project will be of use to
other teachers, especially those who teach Computer Mathematics
1. in the state of Texas.

To obtain the execution speed desired, the author decided to
use a compiled language. The IBM PC (or compatible) was chosen
because of ease of use and availability.

The project required three phases: writing the programs,
writing the project documentation, and compiling test questions.

The author discussed the project with other Computer
Mathematics I teachers and requested their assistance with the
test questions. Those teachers responding are listed in the
bibliography. Also listed are the text books used most often by
the author over the past six years. They are listed because
most of the test questions used by the author on past tests came
from these books or were suggested by those books. Which
questions come from which books the author could not remember
after all this time.

To write the program, the author chose to use Borland's
Turbo Pascal. To learn Turbo Pascal, the author read John H.
Stockman's PROGRAMMING USING TURBO PASCAL. The Turbo Pascal
Reference Manual supplied with the software was used to solve
problems in writing what eventually became two programs. The
author also received assistance from the committee chairperson
and various computer lab assistants at the university.
In the planning stages of this project, there was to have been one program, one test bank disk and the necessary documentation. The author began the program with a menu which allowed the user to select the Test Bank Editor or the Test Generator. With that completed and debugged, the author began writing the procedures for the Test Bank Editor. By the time the Test Bank Editor was almost completed, it became evident that the program, as it was first designed, would be too large for main memory. The author decided to break it into 3 programs: a Main Menu, the Test Bank Editor, and the Test Generator. The main menu procedure was separated from the test bank procedures, and the process was tested.

The author discovered that while the Main Menu program could execute the Test Bank Editor program, the Test Bank Editor program could not execute the Main Menu program.

The author decided to create two separate programs, the Test Bank Editor and the Test Generator, and eliminate the menu program. This was a workable arrangement as the two programs had separate uses and would not normally be used at the same time.

The author developed both programs by writing a main menu procedure with stubs for the main modules. Each stub module was developed, coded and tested. Refinements to all procedures were made as the author discovered new techniques and statements in the language.

After reading about both types of files, and trying short
programs with each, the author selected text files as the best type of file for this project. The deciding factor was the amount of coding required to set up a binary file which would include the different types of records that the author wanted in one file.

The author wanted the programs to be executable from the At prompt. To accomplish this, the completed programs were compiled as .COM files. This way, the user would not need Turbo Pascal to run the programs.

During the time that the programs were being developed, coded, and tested, the author was gathering test questions and sorting them into essential elements.

As soon as the Test Bank Editor was complete, the author began entering the questions for the Test Bank.

The author began writing the documentation for the project when both programs were complete.
ENVIRONMENT

The project was written for an IBM-PC hardware environment, requiring an IBM-PC or compatible, with at least 256K of memory, two disk drives, a monochrome monitor, and a dot matrix printer.

The project was developed primarily on a Tandy 1000SX computer with 384K memory, 2 disk drives, an RGB color monitor, and a Tandy DMP 130A dot matrix printer.

The programs were written using Turbo Pascal.
LOGICAL DESIGN

TEST BANK EDITOR

1. Screen 1
2. Create New Bank
3. Change Directions
4. Add File
5. Update Bank
6. Print Out Bank

TEST GENERATOR

1. Screen 1
2. Create Test
3. Check and Get
4. Check and Save
5. Print Test

[] INDICATES THAT LOWER LEVELS EXIST
(and will be diagramed later)
Create New Bank

2.1
Bank Directions

2.2
Bank Questions

2.1
Bank Directions

2.1.1
Select T Name

2.1.2
Print Dir Info

2.1.3
Input Dir

2.1.4
Display Dir

2.1.5
Compute Q Lines

2.1.6
Save Directions

(*) SEE 3.5

2.2
Bank Questions

2.2.1
Get Questions

2.2.2
Check and Save

(+) SEE 5.5