COMPUTER PERSONNEL RESEARCH

ISSUES AND PROGRESS IN THE '60s

COMPUTER PERSONNEL RESEARCH -ISSUES AND PROGRESS IN THE 60's*

DAVID B. MAYER
IBM – Watson Research Center
ASHFORD W. STALNAKER
Georgia Institute of Technology

MAYER: Good morning, Ash.
STALNAKER: Good morning Dave.
MAYER: Good morning ladies and gentlemen.

We weren't always the ACM Special Interest Group on Computer Personnel Research; in fact, we only became so in November, 1966. Originally we were known as CPRG (Computer Personnel Research Group); founded back in 1962 as a result of some discussions at The RAND Corporation. I guess one might say that Robert Reinstedt was a little lonesome for the company of some other researchers. He had been investigating the problem of programmer selection and discovered that there was indeed very little research on this subject -- except in the hands of a few colleagues in his general professional area, namely, Dallis Perry at the System Development Corporation, Professors Raymond Berger and James Rigney at USC, Jim Tupac at RAND, and Dr. Sherwood Peres who was at the Sandia Corporation. He called these people together and they laid out a charter for CPRG in September 1962 at the American Psychological Association meeting. By then they were well underway towards assembling a test battery to be used as the major component of a nationwide study of programmers. However, they needed help to conduct a national testing program, and thus placed an article in the January 1963 issue of DATAMATION. This aroused sufficient interest to permit calling the First Annual Conference of CPRG in Chicago, June 1963. At this point the group decided upon the final design of a test battery to be administered across the country to as many installations and experienced programmers as they possibly could find. The test battery consisted of the Programmer's Aptitude Test, (better known as the PAT) by Hughes and McNamara; the Strong Vocational Interest Blank; a special trial test named the Test of Sequential Instructions; and some personal background material covering education experience, and so forth. The results of this test battery will be included in the body of our talk, but at this point it suffices to note that we completed it successfully, and we calculated correlations on the data, obtained much information, and initiated the kind of research that CPRG performs.

The initial researchers went on from that point to develop a Programmer's Appraisal Instrument, which is an evaluation device. This was the result of Dr. Sid Fine's work with Robert Dickmann (now SIG/CPR Chairman) at Johns Hopkins University. It was tested there at the Applied Physics Laboratory, and at 24 other installations.

The next step in our research was aimed at advancing the state-of-the-art of interest tests. The Strong Vocational Interest Blank had

* *The first session of the Conference was devoted to a review of SIG/CPR's progress in research since 1962 and some thoughts on problems which still face us. This review was in the form of a dialogue between a computer center manager, David B. Mayer, and a management scientist, Professor Ashford W. Stalnaker.*

been revised (this had nothing to do with CPRG) and, after additional testing, Dallis Perry developed a key for programming as a separate occupation.

In 1966 ACM approached us -- they thought that we were doing so well that we should join the computing fraternity, since originally CPRG was composed primarily of psychologists with only a sprinkling of computer managers. We agreed that joining forces with ACM allowed CPRG's research to be enhanced through broader contacts and outlets. Hence in this survey paper today, we would like to detail some of that research history, and some of the existing issues as we see them at this time.

(At this point, the sample test battery contained in the Appendix was administered to the attendees.)

MAYER: Now that you've gone through a small period of trial, we should tell you a bit about the use of some of these tests in selecting computer personnel.

Table I summarizes the results of the Dickmann survey of 1966 which was reported at the Fourth Annual Conference (1). This table indicates that 483 firms in the United States and another 98 in Canada participated in the survey. In the U.S. 68 percent used tests in some form or another for selection. This corresponds very closely to 72 percent in Canada. The number of programmer-analysts actually employed by these organizations was over 23 thousand in the United States, with another thousand in Canada. The number of people who are needed in the forthcoming year as of the time of this survey was another 25 percent.

Table I also shows the composition of the sample by industry groups.

TABLE I

Type of Organizations in 1966 CPRG Survey

	UNITED STATES	CANADA
Organizations Participating	483	98
Programmer/Analysts Involved	23,636	1,083
Approximate Number Hired Each Year	5,317 (25%)	177 (20%)

	UNITED STATES		CANADA	
	NUMBER	PERCENT	NUMBER	PERCENT
AIRCRAFT INDUSTRY (Industrial, Aerospace)	47	10	3	3
ELECTRONIC INDUSTRY (Industrial, Electrical-Electronic)	35	7	2	2
OTHER INDUSTRY (Petroleum, Metal, Automotive, etc.)	120	25	34	35
FINANCE (Banks, Insurance Companies, etc.)	81	17	21	22
RESEARCH (Non-profits, University Labs, etc.)	90	19	10	10
GOVERNMENT (Federal, State, and City Civil Service)	50	10	8	8
UTILITY AND OTHER NON-MANUFACTURING CONCERNS	60	12	20	20
	483	100	98	100

From Dickmann, Ref. 1

The kinds of programming being performed were basically of four major types - BUSINESS, SCIENTIFIC, SOFTWARE or MILITARY, or combinations of these (Table II). Of them, the largest percentages were in business computations with scientific applications coming in somewhat lower. Military and software programming were small by comparison.

TABLE II

Programming Staff Applications

	UNITED STATES	CANADA
Business	186	58
Business and Scientific	84	11
Business and Scientific and Software	72	6
Scientific	44	6
Scientific and Software	34	1
Business and Software	33	1
Business and Scientific and Software and Military	12	0
Software	6	0
Military	4	0
Scientific and Software and Military	3	0
Business and Military	2	0
Business and Software and Military	1	0
Other	2	1

From Dickmann, Ref. 1

What kind of programmers are there? We classified them at the time the survey was designed into four major categories: a programmer who was essentially a junior or trainee; the second one was called the experienced programmer; a third level we called the system analyst trainee; and the fourth one was termed the experienced systems analyst. Table III answers the question as to what kind of education is demanded by the various institutions or organizations in their hiring practices. In the United States it tended toward having some college training or a degree - over 50 percent of the United States sample, especially for the

TABLE III

Educational Requirements

	PROGRAMMER TRAINEE		EXPERIENCED PROGRAMMER		SYSTEM ANALYST TRAINEE		EXPERIENCED SYSTEMS ANALYST	
	U. S.	CANADA	U. S.	CANADA	U. S.	CANADA	U. S.	CANADA
None Specified	9	14	9	10	7	4	8	7
High School	27	65	19	43	13	32	11	28
Some College	25	3	23	6	12	16	14	10
College Graduate	34	13	35	8	43	9	40	11
Graduate Degree	1	2	1	2	2	7	5	7
Not Reported	4	3	13	31	23	32	22	37
	100%	100%	100%	100%	100%	100%	100%	100%

From Dickmann, Ref. 1

programmer-trainee. This tendency is a little more emphasized as you move up the experience ladder. Canada's educational requirements were somewhat lower, possibly because they do not have as large a college population from which to recruit. In Canada, 65 percent of the programmer trainees had only a high school education and the remainder had some college or above. If you consider the experienced Canadian systems analysts, you will find that 28 percent had a high school education or better. However, 37 percent were not reported, so we can only generalize about the true proportions in this situation. Canada, therefore, is drawing on its resources of personnel in accordance with what they have available.

Tests were used in many of these organizations, but they were used differently depending on whether the firm required much education, or little. Or to put it in reverse, possibly tests were NOT used in many cases, and hence the educational requirement was increased to compensate. Taking one category as an example, the systems analyst trainee, almost 50 percent were required to be college graduates if tests were not used; if a test were used, only 39 percent were required to have college degrees. This pattern repeats itself throughout Table IV. We will not dwell on this except to note that tests are used in many cases in conjunction with educational requirements, but in differing degrees.

What types of tests are used in these two countries? The tests reported used were broken down into four major classifications as

TABLE IV

Comparison of Educational Requirements for Organizations Using Tests in Selection Versus Organizations Not Using Tests

(United States Sample)

	PROGRAMMER TRAINEE		EXPERIENCED PROGRAMMER		SYSTEMS ANALYST TRAINEE		EXPERIENCED SYSYEMS ANALYST	
	NON-TEST	TEST	NON-TEST	TEST	NON-TEST	TEST	NON-TEST	TEST
None Specified	6	10	6	10	5	8	7	8
High School	16	32	6	25	4	17	3	15
Some College	27	24	19	25	10	14	7	17
College Graduate	37	32	53	27	50	39	48	36
Graduate Degree	3	1	4	0	4	1	10	3
Not Reported	11	1	12	13	27	21	25	21
	100%	100%	100%	100%	100%	100%	100%	100%

From Dickmann, Ref. 1

shown in Table V. The first major classification is the general intelligence test with the most commonly used being the Wonderlic Personnel Test -- 60 organizations in the United States and 7 in Canada. Of these organizations 13 had undertaken validation studies of this test. Whether the validation studies consisted of actual on-the-job performance validation or training validation was not indicated in the survey. The other two principal tests used are general intelligence tests.

The second type is the aptitude test, which is being used as if it isolates programming as a separate and special aptitude. The IBM Programmer Aptitude Test, PAT, is by far the most commonly used test in both countries: over 282 organizations in the United States -- approximately 83 percent, and 67 in Canada. Of the remaining aptitude tests, the National Cash Register Test, the Science Research Associates Test Battery and several others were used, but nowhere nearly as widely as the PAT. The Federal Service Entrance Examination is not a true aptitude test - it is a test, I believe, that is given to almost any prospective federal service employee, for many different positions.

The two other types found in the survey were the personality tests and the interest tests. Very few of them are used. Personality tests were being used by only 10 or 15 organizations and very sparingly at that. For example, the Thurstone Temperament Schedule and the Activity Vector Analysis were each used by only three organizations; several others were used in varying degrees.

For the interest tests, the Kuder Preference Record was cited by only two organizations. Interestingly enough to me, neither Strong

TABLE V

Tests Used for Interviewing Programmer Candidates

TEST NAME	FREQUENCY OF USE* UNITED STATES	FREQUENCY OF USE* CANADA	VALIDATION STUDIES (TOTAL)
GENERAL INTELLIGENCE TESTS			
Wonderlic Personnel Test	60	7	18
Thurstone Test of Mental Alertness	12	1	4
Otis Tests (Unspecified)	11	0	5
School and College Ability Tests (SCAT)	5	0	2
Wesman Personnel Selection Test	3	0	0
Ship Destination Test	3	0	0
Lowry Lucier Reasoning Test Combination	3	0	2
Concept Mastery Test	2	0	1
Henmon Nelson Tests of Mental Ability	2	0	2
Schubert General Ability Battery	2	0	0
APTITUDE TESTS AND BATTERIES			
IBM Programmer Aptitude Test	282	67	83
National Cash Register Programming Aptitude Test (E51)	9	4	6
Federal Service Entrance Exam	13	0	3
SRA Computer Programmer Aptitude Battery (Burroughs Corp.)	5	3	1
Employee Aptitude Survey	7	0	3
Differential Aptitude Tests	6	0	4
Watson Glaser Critical Thinking Appraisal	5	1	3
Short Employment Tests	5	0	1
Test of Sequential Instructions	2	0	1
Minnesota Clerical Test	2	0	1
Guilford Zimmerman Aptitude Survey	2	0	1
Bennett Mechanical Comprehension Test	2	0	2
PERSONALITY TESTS			
Thurstone Temperament Schedule	3	1	1
Activity Vector Analysis	3	0	3
Rohrer-Hibler-Replogle Personality Test	2	0	0
Humm-Wadsworth Temperament Scale	2	0	1
Edwards Personal Preference Schedule	2	0	0
Cleaner Self Description	2	0	0
Adaptability Test	2	0	0
Guilford Martin Inventory of Factors	1	0	1
Guilford Martin Temperament Profile Chart	1	0	1
INTEREST TESTS			
Kuder Preference Record	2	0	2
OTHER TESTS			
Manhattan Symbol (MAZE)	4	0	2
1401 Autocoder Exam	3	0	0
GCT	2	0	1
LOMA	2	0	1
Personagraph	2	0	0

*For Tests Used Two or More Times

From Dickmann, Ref. 1

Vocational Interest Blank (SVIB, original or revised version) was mentioned by any organization. Yet it is one of the few for which there exists a key for programming. It is hoped that SIG/CPR will have some educational effect by bringing the value of SVIB to the attention of those responsible for personnel selection.

Among the other tests is the 1401 Autocoder exam, which was developed by Computer Usage Corporation. This test is a form of the Logical Analysis Device developed by Langmuir at the Psychological Corporation of America. This latter test, known as the LAD, was not cited at all, however.

Let us take a look at the use of the PAT for a moment. (See Table VI.) 282 organizations use it in the United States. 128 of them use it in combination with some other test. 154 use it alone. As for position levels at which it is used, for the programmer trainee - 278 organizations gave the PAT; for the experienced programmer -- and this is always a delicate subject for a computer manager who is interviewing candidates -- there were 138 organizations; for systems analyst trainees - 142; and for experienced systems analysts - only 87. Of these, 71 organizations had performed validation studies - that is, how the performance of the programmer compared to his score on the PAT. 22 organizations, or a little less than 10 percent, actually discontinued the use of the PAT for various reasons.

This completes our summary of the Dickmann survey, and I think it would be well worth while to go into some of these tests and describe them and relate them to the sample battery which you have just completed. Ash, would you discuss some of these for us, please?

TABLE VI

1966 CPRG Survey IBM Programmer Aptitude Test

	UNITED STATES	CANADA
NUMBER OF ORGANIZATIONS USING:	282	67
IN COMBINATION WITH OTHER TESTS:	128	18
ALONE:	154	49
PERCENT OF TOTAL SAMPLE:	58%	68%
PERCENT OF THOSE USING TESTS:	85%	93%
LEVELS OF TEST USE:		
PROGRAMMER TRAINEES:	278	67
EXPERIENCED PROGRAMMERS	138	27
SYSTEMS ANALYST TRAINEES:	142	32
EXPERIENCED SYSTEMS ANALYSTS:	87	14
VALIDATION STUDIES:	71	12
DISCONTINUATION:	22	0

From Dickmann, Ref. 1

STALNAKER: Before we begin a discussion of the tests themselves, we should make clear some of the facts about the population to which we are addressing ourselves. Of the studies on job analysis which have been performed relating to computer programmers, there are two which have been cited in SIG/CPR's literature. Before we look at the results of these, I think some remarks from one of the studies are quite appropriate. These observations were made by Ray Berger of the University of Southern California. His first observation was that job titles such as systems analyst, senior programmer and programmer hold only a rough approximation of the job content. Secondly, the designation of content areas such as scientific and engineering, business and logistics, and military systems, serve better to distinguish areas of computer application rather than areas of programming.

Of the two studies that have been cited in the SIG/CPR literature, the first was performed by Lothridge (2) at General Electric Corporation. Lothridge's study consisted of presenting to experienced programmers 40 job statements which they were required to order in rank of importance or in terms of the frequency of occurrence in their jobs. As a result of these orderings, Lothridge was able to develop three job descriptions. These job descriptions are, however, unnamed. In the Berger study (3), a similar approach was taken, except that 186 task statements were given to experienced programmers who ranked them similarly to the method used in the Lothridge study. As a result of this work, Berger was able to develop a list of 17 general programming operations. These are shown on Table VII. If it is agreed that these do in fact represent the 17 major tasks of programming, they do not in my mind give us a satisfactory basis for job analysis or a satisfactory set of job descriptions that adequately describe what a programmer really does at various levels. David, do you feel that these 17 tasks in any way represent what really goes on in the programming world?

TABLE VII

17 Major Tasks of Programming

I. PROGRAM PRODUCTION

1. General Programming Operations
2. Debugging
3. Programming Real-Time Systems
4. Lead Programming Responsibility
5. Program Production; Special Purpose Computers
6. Program Production; Planning and Scheduling
7. Program Production Supervision
8. Utility Program Development (Executive and Compiler)
9. Utility Program Development (General Purpose and Library)
10. Program Diagramming and Testing

TABLE VII

(cont'd)

II. PROGRAM ANALYSIS AND DESIGN

11. Program Systems Analysis (Business & Logistics)
12. Program System Analysis and Design
13. Program System Integration

III. TESTING, INSTRUCTION, DOCUMENTATION AND TRAINING

14. Program System Testing
15. Program Installation or Modification Consulting
16. Program Documentation
17. Training

From Rigney-Berger, Ref. 14

MAYER: Yes, to a large extent, I think they do. However as a computer center manager, I think it would be interesting to compare Berger's list with what managers think really goes on, i.e., a list which they use to rate their employees. It just so happens I have here the supervisory rankings of the 42 items in the Programmer Appraisal Instrument (4). (See Table VIII.) Of the 42 items, the majority of the ten considered most important involve knowledge and capability, such as checking out programs, understanding assignments, and defining problems. Only a few involve the temperament of the person - that is, if he is diligent, and can handle work under pressure. The ones that are ranked lowest on the scale of importance, as far as managers are concerned, are age, the number of professional societies he belongs to and so on.

I think though, more interesting is what programmers think is the content of their job; they decided to be entirely technical. Table IX is from Bairdain's study (5) and displays the patterns of techniques that are used in programming from the programmer's point of view. These range from group A down through group E with increasing complexity, i. e., from a junior trainee who knows how to do looping and instruction modification through the advanced systems analyst and systems programmer who will have programmed inter-connected computers having remote input/output processing. Notice that none of these involves temperament; they're all purely technical items.

Finally, I think even more interesting was when management decided to observe the programmers in their actual work and see what they really did all day long. The results of this study (5), were very telling. The utilization of time, Table X, shows that if indeed a programmer was trying to produce output, the only time he did so was when he was either reading, writing, or recording - and that happens only 27 percent of the time. All the rest of the time he was talking or

TABLE VIII

Programmer Performance Appraisal Items CPRG Research Study 1964

	TYPE OF ITEM	IMPORTANCE INDEX
10 HIGHEST RATED ITEMS		
CHECK OUT PROGRAMS	K/C	74
PLAN PROGRAMS	K/C	73
DEFINE PROBLEMS	K/C	72
UNDERSTAND ASSIGNMENTS	K/C	70
WORK INDEPENDENTLY	WS	69
FINDS APPROPRIATE PROG'G METHODS	WS	68
DILIGENT	TT	67
CAN HANDLE COMPLEXITY	K/C	66
WORK UNDER PRESSURE	TT	65
MASTER ASSIGNMENTS SPEEDILY	WS	65
10 LOWEST RATED ITEMS		
EVALUATE NEW HARDWARE	K/C	34
INITIATE INVESTIGATIONS IN MATH ANALYSIS	K/C	34
PERSONAL APPEARANCE	P/P	33
HANDLE # OF MACHINES	K/C	32
GIVES ON-THE-JOB TRAINING	WS	30
USES MATH. ANAL. METHODS	K/C	29
# PUBLICATIONS, TALKS GIVEN	P/P	17
# PROF. SOC. BELONG TO	P/P	17
AGE	P/P	16
TEACH FORMAL CLASSES	P/P	10

K/C:	Programming Knowledge/Capability
WS:	Working Style
TT:	Temperament Traits
P/P:	Personal/Professional

From Bairdain, Ref. 5

listening, possibly to a manager or possibly to a friend, walking, away, or out. This was based on over 7,000 observations of a group of methods programmers. Oh, by comparison, we have a study of a group of engineers (see Table XI) under the same kind of observational procedure. It shows that the time devoted to activity which would produce an output for the product in question was 45 percent compared with the programmers' 27 percent; I think the figures stand by themselves, as to what programmers really do.

TABLE IX

Patterns in the Use of Programming Techniques

TECHNIQUES			
A.	LOOPING INSTRUCTION (ADDRESS) MODIFICATION COUNTING INDEXING SWITCHES (HARDWARE, DIGIT, etc.) CONTROL CARDS		CHANNELS CONTROL ROUTINES CONVERSIONS SCANS
B.	LOGICAL OPERATIONS LABELS PROGRAMMED INPUT/OUTPUT INTERRUPTS BLOCKING (or DE-BLOCKING)	D.	MULTIPLE PROGRAMMING CHECKPOINT AND RESTART RANDOMIZING TABLE LOOKUP SEARCHES LIST PROCESSING
C.	SORTING MULTIPLE INPUT/OUTPUT	E.	REMOTE INPUT/OUTPUT INTERCONNECTED COMPUTERS

From Bairdain, Ref. 14

TABLE X

Utilization of Time Work Sampling: Programmers in a Data Processing Center Research Study 1964

ACTIVITY	LIST CARD WORK SHEET	BUS	PERS	MEET	TRNG	MAIL MISC DOC	TECH MAN	OPER PROC MISC	PROG TEST	TOTAL
Talk/Listen	4	17	7	3	--	--	--	1	--	32
Talk W/Mgr	--	1	--	--	--	--	--	--	--	1
Telephone	--	2	1	--	--	--	--	--	--	3
Read	14	--	--	--	--	2	2	--	--	18
Write/Record	13	--	--	--	--	1	--	--	--	14
Away/Out	--	4	1	4	6	--	--	--	--	15
Walk	2	2	1	--	--	1	--	--	--	6
Misc's	2	3	3	--	--	1	--	1	1	11
Total	35	29	13	7	6	5	2	2	1	100**

Approximately 70 Personnel were Involved in this Study.

**The Figures in the Body of the Table Show the Percentage of a Work Week Devoted to the Activity.

From Bairdain, Ref. 5

TABLE XI

Utilization of Time Work Sampling: Engineers in a Research Laboratory Research Study 1963

ACTIVITY	PERCENT OF TIME DEVOTED TO ACTIVITY	
INDEPENDENT WORK		34%
– LABORATORY WORK	13%	
– READING	11%	
– CALCULATING	7%	
– THINKING	2%	
– OBSERVING	1%	
WRITTEN COMMUNICATION		11%
– WRITING	9%	
– DICTATING	1%	
– SKETCHING	1%	
VERBAL COMMUNICATION		40%
– WITH ALL OTHERS	24%	
– WITH SUPERVISOR	9%	
– WITH SUBORDINATES	7%	
OTHER ACTIVITIES		7%
– WALKING	4%	
– PERSONAL	3%	
AWAY/NOT RECORDED		8%

From Bairdain, Ref. 5

STALNAKER: I guess then our task in SIG/CPR is to find a method of selecting people who can communicate well within the group. Apparently, this is their major activity.

MAYER: That may be their major activity, but it certainly isn't the major reason they were hired.

STALNAKER: We will come back to this point a bit later. Next, we would like to consider some of the several tests that have been used in various CPRG studies -- a sample of some of these were included in the small test battery which you just completed. In the original CPRG national survey (6), three cognitive devices were included. These were the PAT, the Strong Vocational Interest Blank, and the Test of Sequential Instructions. The first of these, as was indicated by the Dickmann survey, is by far the most popular selection instrument in both the U.S. and Canada. The results of the first CPRG study indicated positive correlations between the PAT score and actual performance only in a small number of cases. In the overall summary of the report, no correlation was found between the PAT and supervisory rankings of performance.

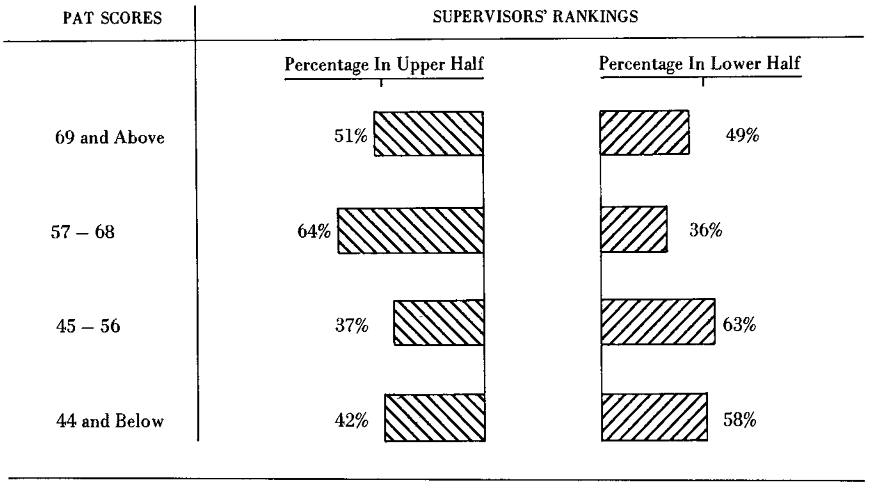
The PAT has also been used in two studies subsequent to the CPRG national survey. The first was by Biamonte (11) at NYU working with a group of non-credit programming students, and the second by Gotterer and Stalnaker (12) at Georgia Tech with several groups of undergraduates enrolled in a computer course which had as a major component programming and systems analysis. In the latter case, as was true in the national survey, no correlation was found between PAT scores and performance in a training situation. We emphasize that the work at Georgia Tech was strictly in a training situation and in no way relates to subsequent performance in an actual programming assignment.

David, what do you think about the matter of a programming aptitude? Does such a thing really exist?

MAYER: Well, I have been trying to find that out from CPRG for several years. You know, after over a thousand interviews -- which I'm told is the worst way to find out whether they are a programmer or have potential -- and approximately 200 people hired over my signature, I would say that I can detect what one might call an X factor; it's probably called aptitude. I do not know of what it consists; all I can say is that this man sitting before me seems to possess it and will succeed in the programming art. After selecting my programmers in this way, when I rank my programmers on some kind of scale from top to bottom, I obtained some very interesting results, which I think we can show later.

STALNAKER: I seriously question that there is, as far as the training situation is concerned, any indication of a specific programming aptitude. I might mention that my interest in CPRG evolved from the same question that I just put to David. The interest was generated primarily by the repeated occasions which we observed at Tech, wherein a truly marginal student -- a student who was only barely able to maintain satisfactory status in school -- was able to succeed in developing a rather sophisticated programming skill and also a sophisticated approach to systems analysis.

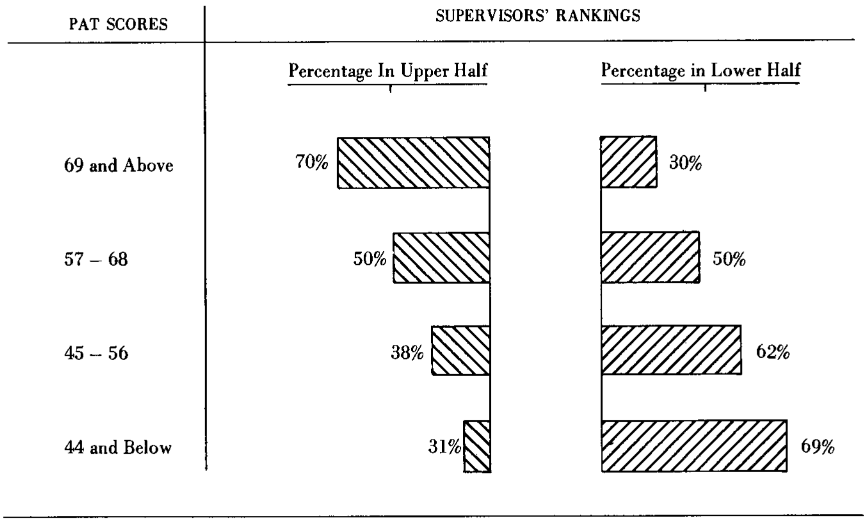
Before going on, David, maybe we should look at some of the CPRG results with regard to the PAT (6). As I mentioned, only in a limited number of cases was there a significant correlation between PAT results and ranked performance. However, I feel that one of the points that we should pay special attention to is the sort of cross-overs or the reversals we get in terms of the PAT. We will notice among the business programmers (Figure 1) who are graded in the upper half with regard to their performance, 42 percent of them scored 44 or below on the PAT. On the other hand, among those who were rated in the lower half in regard to their performances, 49 percent of these scored 69 or above on the PAT. We might note too that the relationship in this case could be curvilinear. In the scientific group, Figure 2, the relationship is approximately linear and the degree of the reversals not as large as in the case of the business group. Here we note that 31 percent of those who are rated in the upper half with regard to the performance scored 44 or below, and 34 percent who scored in the lower half in terms of their performance scored 69 or above. In the sample there were 534 programmers; 301 of them were scientific programmers and the remainder were in business programming. There were 25 different installations and companies.



From Reinstedt, Ref. 6

FIGURE 1

Relationship Between PAT Scores and Rankings – Business Group



From Reinstedt, Ref. 6

FIGURE 2

Relationship Between PAT Scores and Rankings – Scientific Group

A second component of the national study (6) was the Test of Sequential Instructions. The TSI was included to show that <u>any</u> test that in some way measures a form of logical reasoning will in some sense indicate the level of performance in programming jobs and will also indicate in some sense the intelligence of the individual. This hypothesis is borne out by the results of the national study in which the correlations between the TSI and the PAT were in many cases quite significant. One case, David, was yours -- you had some rather remarkable results, didn't you?

<u>MAYER</u>: In my scientific programming group I obtained a correlation coefficient of .70 between the PAT test results and my supervisor's rankings. But interestingly enough, on the TSI the correlation coefficient was .71. So I naturally asked myself, what was I doing that was right? Could I interchange the TSI with the PAT as part of selection procedure for programmers?

<u>STALNAKER</u>: The correlations you obtained on these two tests would indicate to me that they could be used interchangeably in your group to measure the same phenomenon, whatever it was.

<u>MAYER</u>: That strikes me as a little strange. The TSI, to me, tests the ability to do multiple tasks simultaneously. To perform well on that test you are holding one task in the back of your mind while another task is being performed in the foreground. Then, we build up to 3, 4 and 5, in fact, in the real TSI, I think we have 7 tasks running simultaneously. The PAT, to me, has no such attribute. The PAT does test other components which are required in programming -- numerical capability, spatial relationships, and such. I would have suspected the two tests were supplementary rather than interchangeable, yet you are saying that I might be measuring the same phenomenon with two very unlike instruments.

<u>STALNAKER</u>: Again, I remark on the purpose of the TSI -- it was included as a test to measure some form of logical reasoning and thus, will be very closely related to a measure of intelligence. It might well be that PAT is measuring the same thing.

<u>MAYER</u>: Well, in other words, if I hired programmers, they should have these components and they should have some other attributes - I presume, for example, they ought to be interested in the subject.

<u>STALNAKER</u>: This leads us into the third component of the CPRG national the Strong Vocational Inventory Blank (SVIB). The SVIB is a test that has been in use for a great number of years, primarily though, in the vocational counseling area. The purpose of the SVIB is to elicit information regarding the interests of the testee. The interests are then compared to the interests of people who have been successful in several occupational groups, the hypothesis being that if a person has interests similar to those successful in certain occupations, there may be some motivation to enter this occupational area. It should be noted that the SVIB is not claimed by its author to predict performance on the job; instead, it only elicits information regarding interests.

MAYER: You and I have looked at several specific items in the SVIB that are very interesting to me. For instance, the ones about progressive and conservative people seem very telling. Is it possible for me to take individual questions from the SVIB and say that they are definitive of a programmer's attitude, or interest, or something?

STALNAKER: Before answering that question, David, we should compare the interests of computer personnel to the public in general in regard to certain answers to questions on the SVIB (Table XII). The SVIB contains 400 items. You mentioned specifically the matter of progressive people. 41 percent of computer personnel like this type of person -- among the general public 85 percent of these people like people with this outlook. On the other hand, there is a great flocking among computer personnel to conservative people. 84 percent prefer these, while among men in general, only 56 percent. Another example was thrifty people: 45 percent of computer personnel stated they liked this attribute. They are much better liked by people in general -- 74 percent. From what we heard from David's earlier remarks about the work activities of computer personnel, it seems quite reasonable that only 14 percent of the programmers like energetic people whereas 89 percent of the public like energetic people!

TABLE XII

Strong Vocational Interest Blank Interests of Programmers and General Population by Percentages

	PROGRAMMERS			GENERAL POPULATION		
	Like	Indifferent	Dislike	Like	Indifferent	Dislike
Aviator	67%	21%	12%	30%	36%	34%
Mathematics	90	8	3	69	20	11
Solving Mechanical Puzzles	63	29	9	39	34	27
Giving First Aid	22	51	27	40	42	18
Progressive People	41	42	17	85	11	4
Conservative People	84	15	1	56	35	9
Energetic People	14	48	38	89	9	2
Thrifty People	45	44	11	74	22	4
Sell Machines	8	27	65	26	32	42
A — B	Prefer A	Indifferent	Prefer B	Prefer A	Indifferent	Prefer B
Few Details — Many Details	18	27	55	36	28	36
Technical — Supervision	65	16	19	34	19	47

Now, your question -- can we look at individual items in the SVIB to see if these indicate anything in regard to the general interest pattern that should be the pattern of a successful programmer. The answer to this is a decided NO. You mentioned two categories -- progressive and conservative people. The results shown by the SVIB are reversed in work by Biamonte (7) at NYU in which he specifically considered attitudes. He shows there were negative correlations between training success and such attributes as dogmatism, conservatism, and authoritarianism, a finding which would indicate the reverse of your conjecture. The point is that the SVIB questions when taken out of context have no meaning. One must analyze the complete 400 questions in order to elicit anything regarding the total interest pattern of the individual.

MAYER: Then, the procedure is to give the complete test, send it to Minnesota and perhaps two weeks later, I will get some results. This is a difficult way to run an interview.

STALNAKER: It is true - the normal scoring of the SVIB is quite complex because it has to be scored for all occupations. It is possible, I might mention, to hand-score for a particular occupational group. For instance, since the interest of this group is the computer programmer, it could be that the keys could be obtained and thus we could hand-score for this specific occupation. However, I would like to warn against this; the SVIB is meaningful only in terms of its total content. When we take an occupation or a question out of context, the results are questionable to say the least.

I have mentioned the existence of the programmer scoring key for the revised SVIB. The series of questions that you have in your sample battery are from the old SVIB which was the one used in the original CPRG national study. Subsequently, Dallis Perry (8) undertook another national study in which he used the revised SVIB. He also worked with a larger sample than that included in the original CPRG study. Based upon this work, Perry has developed the programmer scoring key which is now available to all users of the SVIB.

MAYER: So far, we have cited aptitude as one component of selection, and we have cited interests as another component. In addition, the popular folklore is that programmers should have very logical minds. Included in today's demonstration battery were some questions from the Watson-Glaser Critical Thinking Appraisal (9). What kind of results should we expect from a critical thinking analysis inventory of this type and should I use it?

STALNAKER: The Watson-Glaser is another quite old test which also has undergone revision since the research which I want to mention briefly. The test is stated by the authors to measure certain critical thinking abilities -- critical thinking abilities which we could hypothesize are, or at least should be, strongly related to programming. The test consists of five parts. As a part of the work I did at Georgia Tech with student programmers the Watson-Glaser was used with four different groups. Even though significant correlations between training performance and Watson-Glaser scores were indicated for three of the groups, we designed a new key and were able to substantially increase the correlations in the last two groups that were tested. Again, though David, I would like to emphasize that the work we have done with this test applies to the student groups and it was used to predict nothing more than training success. Our research says nothing about the tests' relationship to actual on-the-job performance.

MAYER: I gather it was pretty good for predicting training success.

STALNAKER: We think it was quite good as far as predicting training success with respect to the four groups in which it was used.

MAYER: Does it distinguish between a coder and a programmer in any way?

STALNAKER: Well, this is one of the problems which we have yet to face: What kind of programming we're teaching. I don't know how to classify, nor how to judge them on a scale that says a person who does this belongs to that job category.

As I mentioned earlier, the work of Lothridge and Berger in job analysis did not lead to sufficiently detailed job descriptions, by which we can go into the computer personnel population and say you go to job A, and you belong to job C.

MAYER: We have under discussion in the Steering Committee using the test and the modified scoring key in a new national study. I presume that this will be part of a new test battery.

I think now we should move on to the topic of programmer evaluation -- how can we determine if a programmer is actually effective on the job. Ash, would you begin by citing some of the procedures that have been used to analyze performance of the programmer.

STALNAKER: The first research in this area, David, was reported by CPRG. This was the Programmer Appraisal Instrument (4) developed at the Applied Physics Laboratory, under the direction of Bob Dickmann and Sid Fine. This is a multi-dimensional instrument, which in many ways appears to be more concerned with what might be regarded as a professional programmer rather than the operating programmer - at least I think this is sort of a summary of remarks made in the evaluation of the PAI. There is a considerable emphasis on professional activities and this, in some cases, has led to resistance. It is composed of four specific areas: professional preparation and activity, programming competence, dealing with people and adapting to the job. This instrument was validated by Bob Dickmann but is not believed to be widely used at this time.

MAYER: In fact, it is very difficult to use, I don't know if you realize this or not.

STALNAKER: In what sense is it difficult to use?

MAYER: Well, for example, it asks a number of things to be scored numerically, such as: how many societies does he belong to, and how old is he; does he give some on-the-job training -- a lot or none at all? A supervisor finds that this does not really cover the subject of programming, or programming capability. Supervisors shy away from question-and-answer procedures for appraisal of programmers. Practically none have actually put this into effect. Progressive as I am - I haven't either. My project leaders were very resistant to it. They prefer to use subjective techniques, such as their impression of the programmer's output; sometimes they actually read his programs. But a formal document of this type they resist -- it is purely psychological.

STALNAKER: Do you think an instrument such as the PAI really applies to what might be called a coder or junior programmer, since this is the level where correct appraisal is most vital?

MAYER: No, I think a better procedure would be to temporarily take a coder out of his class and put him into the class of programmer, and to observe his programming capability rather than evaluating him through a questionnaire. Or, I would like to have some kind of test which will actually show his level of competence.

STALNAKER: Do you feel that a test that is concerned with programming ability is really going to indicate the readiness for the move, say from coder to programmer, or programmer to senior programmer?

MAYER: It certainly would be more objective than the techniques I have now. There is one further thing that must be done, and that is, as Dr. Paul Herwitz (13) of IBM has recently stated, the only way a supervisor can tell what a programmer is doing is by being knowledgeable about the code that he has written. In other words, he must read the program, and very few supervisors do. That would be the first step I would say towards proper evaluation. The second step would be to give a proficiency test, an objective one.

STALNAKER: In terms of an objective test, Berger (10) at USC, who by the way is a member of the Personnel Electronics Research Group working with the Office of Naval Research on a long-range project, has developed a test which is called The Basic Programming Knowledge Test (BPKT). This test was developed and validated with a group of naval training programmers and also with some outside agencies such as RAND, SDC, etc. Specifically, the test is designed to evaluate six different abilities: first, logic estimation and analysis; second, flow diagramming; third, programming constraints; fourth, coding operations; fifth, program testing and checking; and sixth, documentation. Not only does the test evaluate the person's performance in these areas, but it is also designed to elicit information regarding his basic knowledge of the areas. Do you think this kind of test would serve your purposes, David?

MAYER: Yes; as you know the examples on the last sheet of today's sample battery are from the Berger tests. They are rejected questions, because they did not discriminate between the good programmer and the bad programmer. A set of questions of this kind would be a good proficiency test to my mind. Similar tests for different types of programmers at different levels would be very effective evaluators. The problem of course, is the resistence the programmers are going to show; this applies especially to experienced ones who are very much in demand.

STALNAKER: That would resolve another problem too -- stratifying the various levels of programming.

MAYER: Let me ask you a question at this point. We have cited the PAT, we have cited the Strong Vocational Interest Blank, we have cited the TSI, we have cited Biamonte's attitude survey. The PAT is an aptitude test, presumably. The SVIB is an interests test. The TSI presumably tests a kind of logical capability. Biamonte covers the effects of attitudes. If I use all these tests and get good scores on all of them, does it mean I selected a good programmer?

STALNAKER: No, I don't think it means that you have selected a good programmer - but it may well increase the probability that you have selected one. We have not been able to show at this point, with the exception of the recurring interest pattern of programmer personnel, any strong indication of substantially increasing the probability of correctly selecting a programmer by the use of this battery.

MAYER: Suppose I add the Watson-Glaser using your modified scoring. Then, we add an intelligence test. We have agreed that those who rate high on intelligence tests have made good programmers (maybe it's vice versa, good programmers come from those who are intelligent). Now, if I bring all these scores together, have I selected my programmers properly?

STALNAKER: I think the only thing you have really added new is the Watson-Glaser and of course, it is questionable whether you have really added anything new there. The Watson-Glaser also correlates very highly with intelligence. I would state, probably, David, that if you would use all of your proposed battery to select an individual, you can obtain a person who has a high probability of successfully completing your training program. Whether this individual is going to like programming or will possess the motivation that will allow him to take the successful training onto the job site is a question that is not yet answered.

MAYER: Well, then I think we come to a small denouement, and it is that if we look at the past five years of computer personnel research, the major effort has been in the development of two sets of testing predictors: (Figure 3) testing for training success, and testing for job performance. Then have you said that we have good predictors for the training phase and questionable ones for the working phase? Temperament tests are frequently considered as good predictors of training success and job performance, but we have no research using them.

STALNAKER: We might mention that in terms of temperament tests, even though there is nothing in the public domain regarding these, there

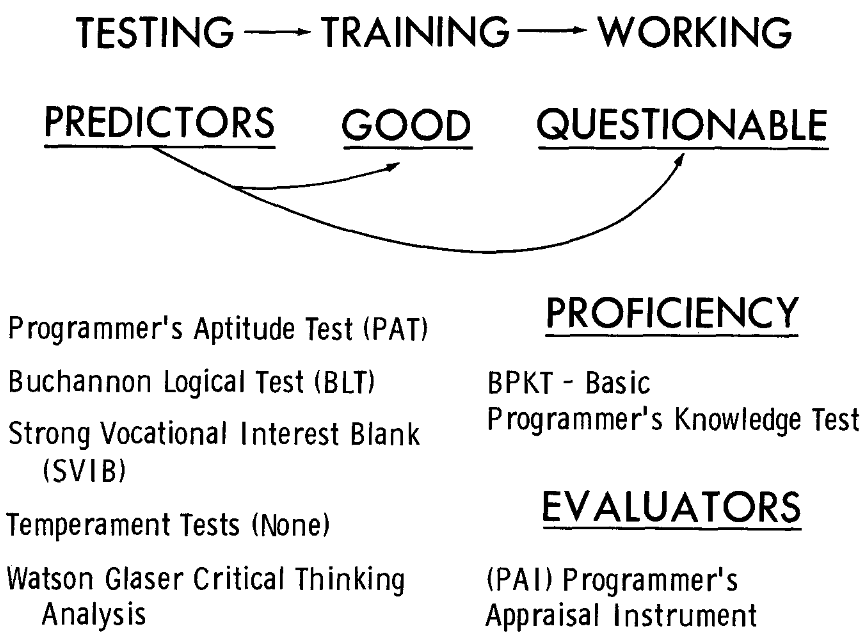

TESTING → TRAINING → WORKING
PREDICTORS
GOOD
QUESTIONABLE
Programmer's Aptitude Test (PAT)
Buchannon Logical Test (BLT)
Strong Vocational Interest Blank (SVIB)
Temperament Tests (None)
Watson Glaser Critical Thinking Analysis
PROFICIENCY
BPKT - Basic Programmer's Knowledge Test
EVALUATORS
(PAI) Programmer's Appraisal Instrument

FIGURE 3

5 Years of Computer Personnel Research

is one study whose rumored results were shocking to say the least. I fully agree with you that temperament is one research area we need to enter.

MAYER: All right, we will leave that an open issue. Finally, of the working tests, the only proficiency test I know of is Berger's Basic Programmer Knowledge Test. I understand that he will publish a version of it in the public domain, by permission of the Navy. For evaluation procedure, at the moment this consists of the Programmer Appraisal Instrument (PAI) which we developed but have not used. This I think sums up the situation at the moment, if I am correct.

STALNAKER: I think that if we have to have a very concise summary of our current knowledge, it is that the more intelligent person you can find, the better programmer you can probably get.

MAYER: That makes all of us here today good programmers!

I think now that we should move from history to the future. Ash and I, of course, discussed this in detail ahead of time. We agreed that there were four major issues -- issues that we feel will be important for computer personnel research to proceed effectively. We have ranked these issues by our estimate of their importance (see Figure 4). Briefly, we think that they stand as follows: The most important single thing we need is an effective evaluation procedure for the programmer as he exists. The second item is the need for a series of selection techniques which recognize the stratified levels of programming -- experienced vs. inexperienced, analyst from programmer; coder from

▲ EFFECTIVE EVALUATION PROCEDURES
▲ STRATIFICATION OF SKILLS
▲ NEW OBSERVATIONAL PROCEDURES
▲ ROLE OF CREATIVITY IN PROGRAMMING

FIGURE 4

Issues in Personnel Research – 1967

programmer, etc. The third issue concerns the need for new observational techniques. Current techniques include testing and interviewing. The fourth issue revolves about the role of creativity in programing. Businessmen always seem to want to hire creative types for their programming staff. Some people feel that programming requires creativity, but there has been very little definitive research in this area.

Let us return to the discussion of evaluation techniques since we have named this as the first and most important issue. We have used several evaluation techniques in CPRG. One of them, for example, has been to rank programmers in a specific order. What ranking was it?

STALNAKER: The order was determined by answering the following question: If the tests were perfectly discriminating, in what sequence from top to bottom would you place your programmers?

MAYER: The trouble with that procedure is that if we rank people in some fashion within an organization, only 25 or so could be ranked at one time within said organization, since that is about all a given supervisor spans. The use of training grades is a standardized approach which can be applied to large groups of students. Effective proficiency tests could be a good spectrum analyzer of hundreds of people. The ones that have been developed in Berger's study have been used in that fashion, but are currently limited to Navy personnel. A final correlate of proficiency which has been used is called "relative salary." What that means is that the salary is adjusted for experience and corrected for geographical and age factors.

STALNAKER: Unfortunately, the hypothesis here is that there is a high correlation between salary and performance -- a very questionable hypothesis.

MAYER: Any subjective procedure will create difficulties in computer personnel research. For one thing, the groups being studied were not homogeneous. For instance, in the national study business groups were ranked separately from scientific groups; but it is impossible to merge two different scientific groups into a single ranking, and likewise, a scientific installation at Lockheed could not be merged with a business group at Johns Hopkins University. We have no way of putting them all together. Furthermore, ranking is relative - it is not an absolute measure. Finally, we could not determine what several different supervisors within an installation thought about the same programmer in more than two or three cases, and you certainly couldn't do it across multiple installations. Hence, evaluation methods must be developed before we can do further effective research, so that we can obtain meaningful correlators across a much wider spectrum and a much more finely divided spectrum. Therefore, the correlators that we need should be national in scope. They should have an effective performance cross-check. They should test knowledge. They should test performance.

STALNAKER: We might mention this point, David, that a test does exist that claims to be in this general area. This is the DPMA Certification program. However, I think we should note that it is quite questionable that there exists a relationship between this test and what we might identify as any level of programming skill. The DPMA test is primarily a knowledge test, but not at a very sophisticated level.

MAYER: Is it knowledge of programming or knowledge of what?

STALNAKER: It is my understanding that it is general knowledge -- at least that is my impression from the groups that were formed to study for it.

MAYER: Then, perhaps the true proficiency test really hasn't been developed yet if this is only a general knowledge test. I would then still call for some kind of standard test, either subjective or objective, but which is replicable. In other words, it can be repeated and the same results can be obtained no matter who gives it, or wherever it is given.

STALNAKER: The second issue that we want to raise is the need for the stratification of skills. All of the work within SIG/CPR, with the exception of Dickmann's survey of test usage, has been concerned with the general category of skill known as programming. We feel that there is a need for research to specifically consider the fact that programming is not a homogeneous skill. There are many different levels of programming and there are many different approaches to programming. We thus raise a question: If we recognize the fact that it is not a homogeneous skill, would it be possible to contemplate a single instrument that could measure the complete spectrum of skills that are required for operating at the various levels within the general area of programming. The specific issue here is first, a study of the contents of the jobs and second, the creation of specific job descriptions. These steps will tend to lead to the stratification that we feel is necessary.

MAYER: Are you saying that only with multiple tests can we obtain multiple stratification?

STALNAKER: Multiple stratification already exists. The question is: Can a single test discriminate between the occupants of these various strata?

MAYER: To summarize: We have called for several new procedures which should be investigated in the years to come. Despite the fact that I have been told time and time again by my psychologist friends that my or other people's interviewing technique cannot be a really effective device, it is still the one that I, as a computer manager, use in at least 50 percent of my evaluation of the candidate. Hopefully, if these new procedures can be developed, both I and my fellow managers will be better able to select, train, evaluate, and reward our computer personnel. Thank you, everyone for your kind attention today.

APPENDIX

SAMPLE TEST BATTERY

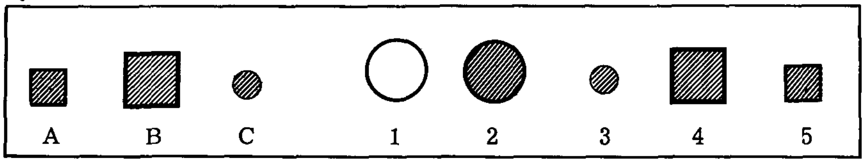
DEMONSTRATION PROGRAMMER'S APTITUDE TEST (DPAT)

	Numerical Series						Circle Next In Series				
1.	2	4	6	8	10	12	(a) 13	(b) 14	(c) 15	(d) 16	(e) 17
2.	1	5	9	13	17	21	(a) 23	(b) 25	(c) 27	(d) 29	(e) 31
3.	14	15	13	16	12	17	(a) 18	(b) 19	(c) 10	(d) 11	(e) 21
4.	3	9	12	9	3	9	(a) 6	(b) 15	(c) 3	(d) 9	(e) 12
5.	12	3	1	2	24	6	(a) 2	(b) 12	(c) 3	(d) 8	(e) 9
6.	9	15	10	20	11	25	(a) 30	(b) 12	(c) 10	(d) 8	(e) 15
7.	1	2	3	4	9	10	(a) 5	(b) 12	(c) 10	(d) 8	(e) 11
8.	4	6	4	8	4	10	(a) 6	(b) 12	(c) 10	(d) 8	(e) 4
9.	100	50	56	28	32	16	(a) 2	(b) 12	(c) 18	(d) 8	(e) 4

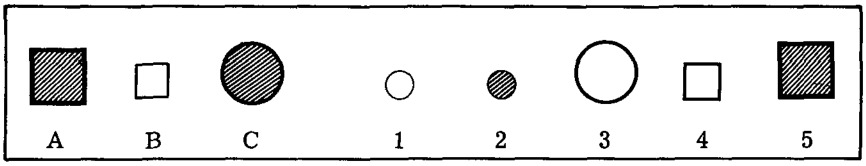
INSTRUCTIONS FOR PART II

Each row is a problem in which A is related to B in some way. You are to find the rule by which A is changed to make B. Then use the same rule to find how C should be changed. One of the numbered figures at the right side of the page is the correct answer.

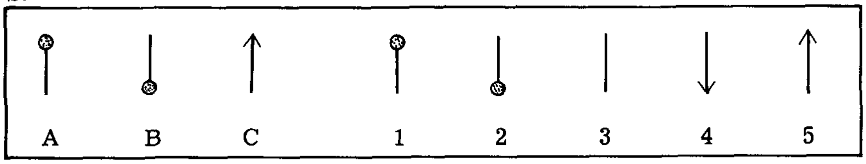
Q.



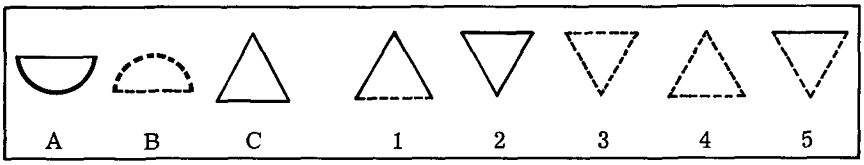
R.



S.



T.



INSTRUCTIONS FOR PART III

After each problem there are five answers, but only one of them is the correct answer. You are to solve each problem and indicate which answer you think is correct by marking the proper space.

X: How many apples can you buy for 60 cents at the rate of 3 for 10 cents ?

(a) 6 (b) 12 (c) 18 (d) 20 (e) 30

Y: In 5 weeks John has saved $3.50. What have his average weekly savings been?

(a) 35¢ (b) 40¢ (c) 50¢ (c) 70¢ (e) 80¢

STRONG VOCATIONAL INTEREST BLANK (SVIB)

INSTRUCTIONS. For the following list of 19 objects, circle the L if you like the item; circle the I if you are indifferent about the item; or circle the D if you dislike the item.

1.	Architect	L	I	D
2.	Airplane Pilot	L	I	D
3.	College Professor	L	I	D
4.	Geologist	L	I	D
5.	Foreign Correspondent	L	I	D
6.	Algebra	L	I	D
7.	Mathematics	L	I	D
8.	Bird Watching	L	I	D
9.	Solving Mechanical Puzzles	L	I	D
10.	Picnics	L	I	D
11.	Sight-seeing Trips	L	I	D
12.	Writing a One-act Play	L	I	D
13.	Giving "First-aid" Assistance	L	I	D
14.	Doing Research Work	L	I	D
15.	Continually Changing Activities	L	I	D
16.	Progressive People	L	I	D
17.	Conservative People	L	I	D
18.	Energetic People	L	I	D
19.	Thrifty People	L	I	D

INSTRUCTIONS. Here are ten things you could do. First read all ten. Then select three things you think you would like most to do, and circle them in the first column next to their numbers. Select the three things you would like least to do, and circle them in the third column. Then circle the remaining four items in the second column where no marks have been made so far.

20.	Develop the theory of operation of a new machine (for example, an auto)	1	2	3
21.	Operate (manipulate) the new machine	1	2	3
22.	Discover an improvement in the design of the machine	1	2	3
23.	Determine the cost of operation of the machine	1	2	3
24.	Supervise the manufacture of the machine	1	2	3
25.	Create a new artistic effect (that is, improve the beauty of the machine)	1	2	3
26.	Sell the machine	1	2	3
27.	Prepare the advertising for the machine	1	2	3
28.	Teach others the use of the machine	1	2	3
29.	Interest the public in the machine through public addresses	1	2	3

(SVIB)

(cont'd)

INSTRUCTIONS. Show here which of two different kinds of work or ways of doing things you like better. If you prefer the items on the left, circle the A column; if you prefer the items on the right, circle in the B column. If you like both the same or if you can't decide which one you like better, circle in the ? column. Work rapidly. Make one mark for each pair.

30.	Taxicab driver	Policeman	A	B	?
31.	Work with few details	Work with many details	A	B	?
32.	Outside work	Inside work	A	B	?
33.	Technical responsibility (in charge of 25 people doing scientific or technical work)	Supervisory responsibility (in charge of 300 people in typical business work)	A	B	?

DEMONSTRATION WATSON - GLASER CRITICAL THINKING APPRAISAL (DWGCTA)

TEST 1. INFERENCE

DIRECTIONS. An inference is a conclusion which a person draws from certain observed or supposed facts. In this test each exercise begins with a statement of facts which you are to regard as true. After each statement of facts you will find several possible inferences - that is, inferences which some persons might make from the stated facts. Examine each inference separately, and make a decision as to its degree of truth or falsity.

In the Answer Area you will find for each inference spaces marked with the letters T, PT, ID, PF, and F. For each inference circle the proper answer as follows:

T	- if you think the inference is definitely true.
PT	- if you think the inference is probably true; that there is better than an even chance that it is true.
ID	- if you decide that there are insufficient data.
PF	- if you think the inference is probably false.
F	- if you think the inference is definitely false.

Sometimes, in deciding whether an inference is probably false, you will have to use certain commonly accepted knowledge or information which practically every person knows.

A thousand eighth-grade students recently attended a voluntary week-end conference in a Midwestern city. At this conference questions of race relations and means of achieving lasting world peace were chosen by the students for discussion, since these were the problems the students felt to be most vital today.

1.	As a group, the students who attended this conference had a keener interest in humanitarian or broad social problems than most eighth-grade students have . . .	T	PT	ID	PF	F
2.	The majority of these students were between the ages of 17 and 18	T	PT	ID	PF	F
3.	The students came from all sections of the country.	T	PT	ID	PF	F
4.	The students discussed only labor relations problems	T	PT	ID	PF	F
5.	Some eighth-grade students felt that discussion of race relations and means of achieving world peace might be worthwhile	T	PT	ID	PF	F

TEST 2. RECOGNITION OF ASSUMPTIONS

DIRECTIONS. An assumption is something supposed or taken for granted. When someone states, "I'll graduate in June," he takes for granted or assumes that he will be alive in June, that he will remain in school until that time, that he will pass his courses, and similar things.

Below are a number of statements. Each statement is followed by several proposed assumptions. You are to decide for each assumption whether it necessarily is taken for granted in the statement.

If you think the given assumption is taken for granted in the statement, circle the appropriate statement.

STATEMENT: "We need to save time in getting there, so we'd better go by plane."

PROPOSED ASSUMPTIONS:

	Assumption	
1. Going by plane will take less time than going by some other means of transportation	Made	Not Made
2. It is possible to make plane connections to our destination	Made	Not Made
3. Travel by plane is more convenient than travel by train	Made	Not Made

----

TEST 3. DEDUCTION

DIRECTIONS. Each exercise below consists of two statements (premises) followed by several proposed conclusions. For the purposes of this test, consider the two statements in each exercise as true without exception. Read the first conclusion beneath the statements, and if you think it necessarily follows from the statements given, answer by circling "CONCLUSION FOLLOWS". If you think it is not a necessary conclusion from the given statements, then circle "CONCLUSION DOES NOT FOLLOW," even though you may believe it to be true from your general knowledge.

Some holidays are rainy. All rainy days are boring. Therefore --

	Conclusion	
1. No clear days are boring	Follows	Does Not Follow
2. Some holidays are boring	Follows	Does Not Follow
3. Some holidays are not boring	Follows	Does Not Follow

TEST 5. EVALUATION OF ARGUMENTS

DIRECTIONS. In making decisions about important questions it is desirable to be able to distinguish between arguments that are strong and those which are weak in so far as the question at issue is concerned.

Strong arguments must be both important and directly related to the question.

Weak arguments may not be directly related to the question, even though they may be of great general importance; or they may be of minor importance; or they may be related to trivial aspects of the question.

Below is a series of questions. Each question is followed by three or four arguments. For the purpose of this test you are to regard each argument as true.

Answer by circling the appropriate answer "STRONG" or "WEAK." When evaluating an argument, judge it on its own merit; try not to let counter-arguments or your own attitude toward the question influence your judgment. Judge each argument separately.

Should all young men go to college?

1. Yes; college provides an opportunity for them to learn school songs and cheers . . . . Strong Weak

2. No; a large percent of young men do not have enough ability or interest to derive any benefit from college training . . . . Strong Weak

3. No; excessive studying permanently warps an individual's personality . . . . Strong Weak

97. Indirect addressing is primarily a method of:

    1. calculating
    2. program-hardware communication
    3. testing core storage
    4. data referencing

102. During the preliminary check-out of a system involving several integrated programs, a programmer should:

    1. run live data through the integrated system to see if expected values appear
    2. use simulated data and check for end results first, then consider the individual programs
    3. use simulated data and consider the individual programs first, then integrate them for end results
    4. use live data and check the individual programs first, then check the whole system

95. Radix sorts are:

    1. best suited for a large amount of data with lengthy keys
    2. best suited for a small amount of data with lengthy keys
    3. fast but require more bookkeeping than other methods
    4. best suited for a large amount of data with short keys.

DEMONSTRATION TEST OF SEQUENTIAL INSTRUCTIONS (DTSI)

		Game #		Total
Team	1	2	3	3 Games
Abel's	390	420	476	1286
Baker's	419	501	427	1347
Charley's	289	394	325	1006

In	this	exercise
1A	1B	1C

each	word	has	a	code
1D	1E	2A	2B	2C

letter	and	number
2D	2E	3A

combination	beneath	it.
3B	3C	3E

For	example,	the	word	"has"	in	the	previous	sentence	has
3E	4A	4B	4C	4D	4E	5A	5B	5C	5D

the	code	2A	beneath	it.	In	the	first	sentence	of	this
5E	6A	6B	6C	6D	6E	7A	7B	7C	7D	7E

paragraph,	circle	the	code	combination	of	the	first	occurrence
8A	8B	8C	8D	8E	9A	9B	9C	9D

of	the	word	"code";	you	should	have	circled	2C,
9E	10A	10B	10C	10D	10E	11A	11B	11C

thus:	code.
11D	2C

Now;	beginning	with	the	next	sentence,	circle	the
12A	12B	12C	12D	12E	13A	13B	13C

code	letters	of	all	words	beginning	with	the	letter	"w."
13D	13E	14A	14B	14C	14D	14E	15A	15B	15C

And	also,	every	time	a	sentence	begins	with	any	vowel
15D	16A	16B	16C	16D	16E	17A	17B	17C	17D

except	"a",	circle	its	code	word.	In	the	meantime	if	the
17E	18A	18B	18C	18D	18E	19A	19B	19C	19D	19E

number	15	is	greater	than	the	number	25,	circle	the	code
20A	20B	20C	20D	20E	21A	21B	21C	21D	21E	22A

letters	of	the	third	word	in	the	previous	sentence;
22B	22C	22D	22E	23A	23B	23C	23D	23E

otherwise,	use	the	third	word	in	this	sentence.	You	may
24A	24B	24C	24D	24E	25A	25B	25C	25D	25E

halt	your	search	for	sentences	beginning	with	"u",	as	well
26A	26B	26C	26D	26E	27A	27B	27C	27D	27E

as	"a."
28A	28B

Team	1	Game # 2	3	Total 3 Games
Abel's	390	420	476	1286
Baker's	419	501	427	1347
Charley's	289	394	325	1006

Observe the bowling
30A 30B 30C

scores of the three-man
30D 30E 31A 31B 31C

teams in the box at
31D 31E 32A 32B 32C

the top of the page.
32D 32E 33A 33B 33C

If Baker's team's third game was his second highest,
33D 33E 34A 34B 34C 34D 34E 35A 35B

then circle the code under the second occurrence of the
35C 35D 35E 36A 36B 36C 36D 36E 37A 37B

word "the" in this paragraph; otherwise circle code "36C."
37C 37D 37E 38A 38B 38C 38D 38E 39A

You may now halt looking for words beginning with the
39B 39C 39D 39E 40A 40B 40C 40D 40E 41A

letter "w" at the end of this sentence. And you may
41B 41C 41D 41E 42A 42B 42C 42D 43A 43B 43C

also halt looking for all those vowels, beginning with the
43D 43E 44A 44B 44C 44D 44E 45A 45B 45C

next sentence. And for your last tasks, if Abel's team
45D 45E 46A 46B 46C 46D 46E 47A 47B 47C

score total was higher than Baker's total then circle codes
47D 47E 48A 48B 48C 48D 48E 49A 49B 49C

"40A" and "42A." Otherwise, circle "40B" and "42B" unless
49D 49E 50A 50B 50C 50D 50E 51A 51B

Charley's second lowest game was less than Abel's lowest
51C 51D 51E 52A 52B 52C 52D 52E 53A

game, in which case circle the code underneath all words
53B 53C 53D 53E 54A 54B 54C 54D 54E 55A

with an "n," in this sentence.
55B 55C 55D 55E 56A 56B

ANSWERS TO DEMONSTRATION QUESTIONS

DPAT

I. Numerical Series

1. (b) = 14
2. (b) = 25
3. (d) = 11
4. (e) = 12
5. (a) = 2
6. (b) = 12
7. (e) = 11
8. (e) = 4
9. (c) = 18

II. Spatial Relations

Q. 2
R. 1
S. 4
T. 3

III. Algebra

X. (c) = 18
Y. (d) = 70¢

DWGCTA

I. Inference

1. PT
2. PF
3. ID
4. F
5. T

II. Assumptions

1. Assumption made
2. Assumption made
3. Assumption not made

III. Conclusions

1. Does not follow
2. Follows
3. Does not follow

V. Arguments

1. Weak
2. Strong
3. Weak

DBPKT

97. Difficulty Level: Easy
Discrimination Index: Unsatisfactory
Answer: 4. data referencing

102. Difficulty Level: Moderate
Discrimination Index: Unsatisfactory
Answer: 3. use simulated data for individual programs, then integrate results

95. Difficulty Level: Difficult
Index: Unsatisfactory
Answer: 2. small amount of data, lengthy keys

DTSI

1. 2C
2. 18E
3. 19A
4. 19C
5. 23A
6. 24E
7. 27E
8. 30A
9. 33D
10. 34D
11. 36C
12. 37C
13. 40C
14. 41C
15. 50E
16. 51B
17. 51D
18. 52D
19. 53C
20. 54D
21. 55C
22. 55D
23. 55E
24. 56B

STRONG VOCATIONAL INTEREST BLANK*

INTERESTS OF PROGRAMMERS BY PERCENTAGES

		Like	Indifferent	Dislike
1.	Architect	72	21	7
2.	Aviator	67	21	12
3.	College Professor	62	26	12
4.	Explorer	67	20	13
5.	Foreign Correspondent	55	27	18
6.	Algebra	90	8	3
7.	Mathematics	90	8	3
8.	Observing Birds	22	37	41
9.	Solving Mechanical Puzzles	63	29	9
10.	Picnics	71	25	4
11.	Excursions	71	25	4
12.	Vaudeville	44	39	17
13.	Giving First Aid	22	51	27
*14.	Doing Research Work	70	23	7
15.	Continually Changing	62	29	9
16.	Progressive People	41	42	17
17.	Conservative People	84	15	1
18.	Energetic People	14	48	38
19.	Thrifty People	45	44	11

		Like	Indifferent	Dislike
20.	Develop the Machine	63	26	11
21.	Operate the Machine	27	43	30
22.	Improve the Machine	64	32	3
26.	Sell the Machine	8	27	65
28.	Teach Use of the Machine	46	42	12

	A - B	Prefer A	Indifferent	Prefer B
30.	Chauffeur - Chef	37	19	44
31.	Few Details - Many Details	18	27	55
32.	Outside Work - Inside Work	29	33	38
33.	Technical - Supervisory	65	16	19

*See *RM 4033-PR, March 1964,* Computer Personnel Research Group Prediction Study, *by Reinstedt, et al, Rand Corp., pp. 43ff.*

REFERENCES

1. Dickmann, R.A., "A Survey of Computer Personnel Selection Methodology," *Proceedings of the Fourth Annual Computer Personnel Research Conference,* pp. 15-27.

2. Lothridge, Charles, "Levels of Classifying Data-Processing Personnel," *Proceedings of the Second Annual Conference Computer Personnel Research Group,* pp. 13-28.

3. Berger, R.M. and Wilson, R.C., "The Development of Programmer Evaluation Measures," *Proceedings of the Third Annual Computer Personnel Research Conference,* pp. 6-17.

4. Dickmann, R.A., "A Programmer Appraisal Instrument," *Proceedings of the Second Annual Conference Computer Personnel Research Group,* pp. 45-64.

5. Bairdain, E.F., "Research Studies of Programmers and Programming," IBM Corporation, 1964, pp. 78, 136, 62.

6. Reinstedt, R.N., et al., *Computer Personnel Research Group Programmer Performance Prediction Study (RM-4033-PR),* The RAND Corporation, Santa Monica, California, March 1964.

7. Biamonte, A.J., "A Study of the Effect of Attitudes on the Learning of Computer Programming," *Proceedings of the Third Annual Computer Personnel Research Conference,* pp. 68-74.

8. Perry, D.K., and Cannon, W.M., "A Vocational Interest Scale for Computer Programmers – Final Report," *Proceedings of the Fourth Annual Computer Personnel Research Conference,* pp. 61-82.

9. Stalnaker, A.W., "The Watson-Glaser Critical Thinking Appraisal as a Predictor of Programming Performance," *Proceedings of the Third Annual Computer Personnel Research Conference,* pp. 75-78.

10. Berger, R.M. and Wilson, R.C., "Correlates of Programmer Proficiency," *Proceedings of the Fourth Annual Computer Personnel Research Conference,* pp. 83-95.

11. Biamonte, A.J., "Predicting Success in Programmer Training," *Proceedings of the Second Annual Conference, Computer Personnel Research Group,* pp. 9-12.

12. Gotterer, M. and Stalnaker, A.W., "Predicting Programmer Performance Among Non-preselected Trainee Groups," *Proceedings of the Second Annual Conference Computer Personnel Research Group,* pp. 29-44.

13. Herwitz, P.S., "Programmer Evaluation," *GUIDE Organization Talk, Management and Administration Committee,* 1966 (IBM, Armonk, N.Y.), pp. 1-12.

14. Rigney, J.W. and Berger, R.M., "Computer Personnel Selection and Criterion Development: II. Description and Classification of Computer Programmer and Analyst Jobs," *Tech. Rpt. 37, Proj. NR 153-093,* Dept. of Psych., U. of Southern Calif., Dec. 1963, pp. 26 ff.