

**APTITUDE FOR  
PROGRAMMING  
PRACTICE**

## APTITUDE FOR PROGRAMMING PRACTISE EXAMPLES

## INSTRUCTIONS

The following practise examples have been provided for you as reasonable example of what to expect on attending your Aptitude for Programming Test. *Please note that no further advice or support will be provided.*

These instructions are presented in a similar format to those that you will receive in the actual test. On the day you will have 1 hour to complete the test. Please note that the test will be a written test, not on a computer. The test itself and these example questions are designed to assess your aptitude for computer programming. It is not an intelligence test and equally it is *not* testing existing programming knowledge.

As such, although you may recognise some similarities between the programming language used in these examples and other programming languages the example language is basically very different. If you have had some experience in computer programming, try to set it aside and use only the exact instructions given here. The same will apply when you attend the test – the language used there will be different again.

You must use the commands in the way defined in the rest of this booklet. You should not place any different meaning on these commands other than that given explicitly in this document. The object of the questions is to determine how well you can utilise the instructions given to solve various types of problems.

The questions all require you to write out the list of programming commands that will solve the problem.

You may only use the numbers and variable names given in the question. In some questions you will need to store a value half way through a program and then reintroduce it into the program. In these cases an intermediate storage variable will be indicated in the question.

If you are given an arithmetical equation to program, then you must reproduce that equation in the list of commands that you write. The following rules apply to the order of calculation:

- Where brackets are indicated, work out what is in the brackets first.
- If one expression divides another, then calculate the bottom line first.
- Calculation order is left to right in the equation.

Following the initial examples are some arithmetic problems to program. Use the commands as outlined in the Instructions. For each problem you should write out the list of programming commands that will solve the problem. These questions have been provided to give you some practice prior to the test. As such, we advise that you try to work through the questions attempting all before looking at the answers, which are provided separately at the end of this document.

## LIST OF COMMANDS

In the following list, A is used as an example of a variable that can be used with each command. When you are writing programs with this language you may use any of the variables in the question or a numeric value given in the question.

All calculations are performed in a working area. This is a temporary storage space and can hold just one value at any time. Where brackets are indicated they must be used.

**BEGIN** – indicates that the program is starting. BEGIN must be in every program and can only occur on the first line of the program.

**PUT(A)** – puts the value of the variable A into the working area. The variable A is unchanged by the command PUT.

**PLUS(A)** – adds the value of the variable A to the value in the working area. The result is kept in the working area. The variable A is unchanged.

**MINUS(A)** – subtracts the value of the variable A from the value in the working area. The result is kept in the working area. The variable A is unchanged.

**TIMES(A)** – multiplies the value in the working area by the value of variable A. The result is kept in the working area. The variable A is unchanged.

**OVER(A)** – divides the value in the working area by the value of the variable A. The result is kept in the working area. The variable A is unchanged.

**SET AS A** – Sets the variable A to the value currently in the working area. Any previous value in variable A is overwritten. The value in the working area remains unchanged.

**CLEAR** – The workspace is cleared of any value currently held and then set to zero. The workspace must be cleared at the end of each program. The CLEAR command can also be used at any stage in any program and any number of times.

**COMPARE (condition, line number)**

– COMPARE is followed by a condition, e.g.  $X=Y$ . If the condition is true at the time when the COMPARE command is encountered then the program moves to next line and continues. If the condition is false then the program moves to the line number given. The condition must be of the form:

A EQUAL B                   – is true when the values of A and B are equal  
 or    A GREATER B       – is true when A has a greater value than B  
 or    A LESS B            – is true when A has a value less than that of B

The condition and line number must appear in the brackets exactly as

above:

– Separated by a comma  
 – The condition first, the line number second

e.g. COMPARE(A EQUAL B, 10)

**CYCLE A START** – repeats the sequence of commands enclosed between the CYCLE START and CYCLE FINISH. A defines the number of times the sequence is repeated. A is optional, i.e. you need not include it in after CYCLE; however, if A is not written then the sequence repeats itself indefinitely. A condition exit from the CYCLE START ... CYCLE FINISH command can be affected by incorporating a COMPARE (condition, line number) statement immediately following the CYCLE START statement. The line number of the COMPARE statement can direct the program to any line inside or outside the CYCLE START ... CYCLE FINISH.

**CYCLE FINISH**

EXAMPLES

EXAMPLE 1

$$X + Y = Z$$

The solution is:

EG1	
1	BEGIN
2	PUT(X)
3	PLUS(Y)
4	SET AS Z
5	CLEAR

Note that the problem is broken down into 3 units:

1 X  
2 +Y  
3 =Z

Each is tackled separately and is given a line of code to represent it. Note that the question is tackled as it is read from left to right. Although  $X + Y$  is arithmetically the same as  $Y + X$  reversing the first two lines of the program would not be exactly reproducing the equation and therefore would be marked wrong.

**EXAMPLE 2**

$$\frac{(U - V)}{\quad} = Z$$

W

The solution is:

EG2	
1	BEGIN
2	PUT(U)
3	MINUS(V)
4	OVER(W)
5	SET AS Z
6	CLEAR

**EXAMPLE 3**

Add X to Y, 3 times and then add W. Store in V.

The solution is:

EG3	
1	BEGIN
2	PUT(Y)
3	CYCLE 3 START
4	PLUS(X)
5	CYCLE FINISH
6	PLUS(W)
7	SET AS V
8	CLEAR

Practise Questions

1.	$\frac{(7 \times 3) + 5}{Q} = X$
----	----------------------------------

2.	Divide 72 by 6. Subtract 7 and multiply the result by 23. Store as F
----	--

1	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

2	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

3. Subtract 3 from Z, the number of times given by the value of D.

4. The following program will not work as written. However, it will work if just one line of the code is different. Write the one line correction to the program.

Multiply the sum of  $6 + 7$  by 16 and store in P.

1	START
2	PUT(6)
3	PLUS(7)
4	TIMES(16)
5	SET AS P
6	CLEAR

3	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

4	
Line	Command

5.	Add 1 to the variable X until it equals the value in the variable Z. Then stop.
----	---

6.	A teaboy has to make sure that there are more than C biscuits in the biscuit tin. He checks the biscuit tin every tea break. If there are more than C biscuits he takes X biscuits out to distribute. If B is the number of biscuits currently in the tin, write a program to calculate N (which has a starting value of zero) the number of tea breaks the biscuit tin lasts.
----	--

5	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

6	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

**Solutions**

The following are suggested answers for the practise questions. For certain questions there may be more than one correct answer. In these cases the various options are listed.

**Question 1:**

Q1	
1	BEGIN
2	PUT(7)
3	TIMES(3)
4	PLUS(5)
5	OVER(Q)
6	SET AS X
7	CLEAR

**Question 2:**

Q2	
1	BEGIN
2	PUT(72)
3	OVER(6)
4	MINUS(7)
5	TIMES(23)
6	SET AS F
7	CLEAR

Question 3:

Q3	
1	BEGIN
2	CYCLE D START
3	PUT(Z)
4	MINUS(3)
5	SET AS Z
6	CYCLE FINISH
7	CLEAR

	Option 2
1	BEGIN
2	PUT(Z)
3	CYCLE D START
4	MINUS(3)
5	SET AS Z
6	CYCLE FINISH
7	CLEAR

Question 4:

Q4	
1	BEGIN
2	PUT(6)
3	PLUS(7)
4	TIMES(16)
5	SET AS P
6	CLEAR

Question 5:

Q5	Option 1
1	BEGIN
2	CYCLE START
3	COMPARE (X LESS Z, 8)
4	PUT(X)
5	PLUS(1)
6	SET AS X
7	CYCLE FINISH
8	CLEAR

	Option 2
1	BEGIN
2	PUT(X)
3	CYCLE START
4	COMPARE (X LESS Z, 8)
5	PLUS(1)
6	SET AS X
7	CYCLE FINISH
8	CLEAR

Q5	Option 3
1	BEGIN
2	PUT(X)
3	PLUS(1)
4	SET AS X
5	COMPARE (X EQUAL Z, 2)
8	CLEAR

Question 6:

Q6	Option 1
1	BEGIN
2	CYCLE START
3	COMPARE (B GREATER C, 10)
4	PUT(B)
5	MINUS(X)
6	SET AS B
7	PUT(N)
7	PLUS(1)
8	SET AS N
9	CYCLE FINISH
10	CLEAR

	Option 2
1	BEGIN
2	PUT(B)
3	MINUS(X)
4	SET AS B
5	PUT(N)
6	PLUS(1)
7	SET AS N
7	COMPARE (B LESS C, 2)
8	CLEAR