# MULTIFA – Multiprecision factorial



**Abstract**

*MULTIFA is a simple program written in 1980 for the SHARP PC-1210, PC-1211 and PC-1212 pocket computers and compatibles to compute all the digits of the factorial of a given integer using multiprecision. Two detailed examples are given.*

***Keywords:*** *factorial, multiprecision, SHARP pocket computers, PC-1210, PC-1211, PC-1212, compatibles*

## 1. Introduction

*MULTIFA* is a simple 5-line *(~200-byte)* practice program I wrote in 1980 for the *SHARP PC-1211* pocket computer and compatible models (runs *as-is* in the *SHARP PC-1210* and *PC-1212* and with minimal or no changes in other models), which can compute all the digits of the exact factorial of a given integer *( N! )* using multiprecision multiplications.

## 2. Program Listing

```
10: "A" CLEAR: H=31,M=H,G=10000000,F=G-1: INPUT "N=";N: A(H)=1
20: FOR Y=2 TO N: FOR Z=H TO M: A(Z)=YA(Z): NEXT Z: FOR Z=H TO M
30: IF A(Z)>F LET L=Z+1,K=A(Z)/G,A(L)=A(L)+INT K,A(Z)=G*(K-INT K): IF M<L LET M=L
40: NEXT Z: NEXT Y; BEEP 1: USING "##.########": IF A(M)=0 LET M=M-1
50: FOR Z=M TO H STEP -1: PRINT A(Z)/G: NEXT Z
```

## 3. Usage Instructions

See the worked examples to understand how to use the program.

## 4. Examples

The following examples can be used to check that the program was correctly entered and to understand its usage.

### *4.1 Example 1*

Compute the exact value of **20!**

In **DEF** Mode, proceed as follows:

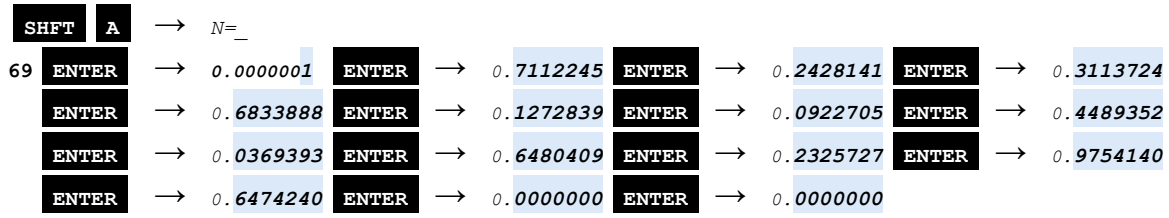
**SHFT** **A** → *N=_*
20 **ENTER** → *0.00**24329*** **ENTER** → *0.**0200817*** **ENTER** → *0.**6640000*** **ENTER** → *>*

So the exact *19-digit* value is: ***20! = 24329 0200817 6640000***

*4.2 Example 2*

Compute the exact value of **69!**

In **DEF** Mode, proceed as follows:




So the exact *99-digit* value is:

***69! = 1 7112245 2428141 3113724 6833888 1272839 0922705 4489352 0369393 6480409 2325727 9754140 6474240 0000000 0000000***

You may want to verify that the program correctly produces this *206-digit* exact value (though it takes a while):

**123! = 121 4630436 7025329 6757662 4324188 1295855 4542170 8848338 2315328 9181618 2923589 2362167 6688311 5696061 2640202 1707358 3522129 4047782 5910915 7041165 1472186 0295199 0626164 6730733 9074198 1495296 0000000 0000000 0000000 0000000**

**Notes**

*1*. The program considers the multiprecision ongoing product as subdivided into *7-digit blocks*, one block per element of array *A*. Each 7-digit block is multiplied by *k* (2,3,...,*N*) and then carries are released after all blocks have been so processed.

*2*. Each block is 7-digit long so as to allow for factorials of *up to 3-digit N*, i.e. up to *N=999*, memory permitting. Multiplying a 7-digit number times a 3-digit number gives a 10-digit result at most, so overflow is avoided. If *N* were restricted to 2 digits, i.e.: up *to N=99*, then 8-digit blocks could be used instead and both memory usage and computation time would be reduced.

*3*. In order to show intermediate *0's* correcty, each 7-digit block is divided by $\mathbf{10^7}$ before outputting it (thus rendering it as *0.ddddddd* where all the *d's* are significant and can be ***0*** ), lest the leftmost digits of a block would not be output if zero. This works for all blocks, but the very *first one* usually can have non-significant *0's*, e.g.*: 0.0000121* is *0000121*, i.e. just ***121***.

**References**

None.

**Copyrights**