Integer Factorization with User Input for the Casio fx-5800P Calculator

https://github.com/slugrustle/fx-5800P_progs Version 1

Program **WFSUB**

```
E+1→E:
B→Z[E]:
Do:
   D→A:
   Z[E+11]+1→Z[E+11]:
   A÷B→D:
LpWhile Frac(D)=0:
Int(√(A))→C:
Return
```

Program Outline

Lines 1–2: Increment number of unique factors and store B as a unique factor.

Lines 3–7: Divide out powers of B from A, incrementing the exponent of B in factor storage.

Variable Descriptions

A: Unfactored part of original input number.

B: Known factor of A.

C: Largest remaining factor in A, assuming A is composite.

D: Temporary storage for A÷B.

E: Number of stored unique factors (ignoring powers). E∈[1,11].

Program **FACTOR**

```
0→DimZ:
22→DimZ:
"NUMBER"?→F:
If F<1 Or F≥1X10 10:
   Then "NUMBER MUST BE  ≥1 And <1X10 10":
   Stop:
IfEnd:
If F≠Int(F):
   Then "NUMBER MUST BE  AN INTEGER":
   Stop:
IfEnd:
For 1→E To 22:
   0→Z[E]:
Next:
1→Z[1]:
1→Z[12]:
1→E:
F→A:
Int(√(A))→C:
2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
3→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
5→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
7→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
11→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
While 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
```

```
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+8→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+8→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+6→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+4→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+10→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+2→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
   B+10→B:A÷B→D:Frac(D)=0⇒Prog "WFSUB":B>C⇒Goto 1:
WhileEnd:

Lbl 1:
If A>1:
   Then E+1→E:
   A→Z[3]:
   1→A:
   1→Z[E+11]:
IfEnd:
Int(E÷3)→D:
E−3×D>0⇒D+1→D:
1→C:
Lbl 2:
Cls:
Locate 1,1,F:
Locate 12,1,C:
Locate 13,1,":":
Locate 14,1,D:
3×(C−1)+1→B:
Locate 1,2,Z[B]:
Locate 11,2,"^(":
Locate 13,2,Z[B+11]:
Locate 16,2,")":
If B+1≤E:
   Then Locate 1,3,Z[B+1]:
   Locate 11,3,"^(":
   Locate 13,3,Z[B+12]:
   Locate 16,3,")":
IfEnd:
If B+2≤E:
```

```
   Then Locate 1,4,Z[B+2]:
   Locate 11,4,"^(":
   Locate 13,4,Z[B+13]:
   Locate 16,4,")":
IfEnd:
While 1:
   GetKey→A:
   If A=34 Or A=73:
      Then Cls:
      "DONE":
      Stop:
   IfEnd:
   If A=84 Or A=86 Or A=77 Or A=47:
      Then C+1→C:
      C>D⇒1→C:
      Goto 2:
   IfEnd:
   If A=83 Or A=85 Or A=67:
      Then C−1→C:
      C<1⇒D→C:
      Goto 2:
   IfEnd:
WhileEnd
```

Program Outline

Lines 1–2: Set up memory for storing 11 factors in the form Z[α]^Z[α+11] where α∈[1,11].

Lines 3–11: User input of number to be factored.

Lines 12–17: Zero out factor storage and store 1^1 as the first factor.

Lines 18–81: Wheel factorization of input with a basis of 2, 3, 5, and 7.

Lines 82–125: Display factors and original input number.

Variable Descriptions

A: Unfactored part of original input number and identifier of the most recently pressed key.

B: Known factor of A and index of displayed factor.

C: Largest remaining factor in A (assuming A is composite) and number of displayed factor page.

D: Temporary storage for A÷B and number of factor display pages.

E: Number of unique factors in factor storage (ignoring powers). E∈[1,11].

F: Number to be factored (input by user).

Notes

There is only storage for 11 unique factors because 6469693230 = 1×2×3×5×7×11×13×17×19×23×29 has the most unique factors of any number on the range [1,1x10^10].

Line 4: The fx-5800P can only represent numbers on the range [−1x10^10,1x10^10] as exact integers.

Lines 5 and 9: The weird spacing prevents text wrapping from occurring in the middle of a word.

Line 110: Pressing DEL (34) or EXIT (73) ends the program.

Line 115: Pressing ▲ (84), ▶ (86), + (77), or EXE (47) cycles to the next factor display page.

Line 120: Pressing ◀ (83), ▼ (85), or − (67) cycles to the previous factor display page.