INVENTORS EMPORIUM

HP41CX Program Listing

Title:	Stamp Selector
Program Label:	LBL "PO" / 82 bytes & LBL "UPO" / 84 bytes
Version:	1.1
Date:	20/02/1989

Description

When I worked in our family business, sending the post was a chore because I had to calculate the stamps required from a book of different values after weighing each letter or parcel. This program determines the fewest number of stamps that will add up to the total postage value you require.

The program uses an extended memory file to keep a list of the stamps you actually have and uses values only from this list. A separate program called **UPO** updates the file with the stamp inventory. Both are described here.

Core Logic

Before running **UPO** (Update **PO**), you need to create an extended memory data file called "**PR**" in which to store the stamp values which will be read each time you run the **PO** program. The size of the initial file is not important as **UPO** will build a new file when it saves.

Create the file by entering **1** in the X reg, **PR** in the Alpha reg. and executing **CRFLD**.

UPO

The program prompts you within a loop for each stamp value in descending order, storing each in a separate register, and ending with zero, which is the value that exits the loop.

The number of the last register is used to define a block of registers to save to the extended file **PR**.

PO

The program recalls the data from the extended file and writes it into a block of main memory registers, the size of which is based on the number of records.

Values from this list are then recalled in turn and compared to the postage value. If it's less, it's added to the string of stamp values and subtracted from the postage total until the total reaches zero. The trick is not to increment the counter recalling the stamp values until it's larger than the postage value remaining.

Execution

UPO

If you have not already done so, Create the extended file by entering **1** in the X reg, **PR** in the Alpha reg. and executing **CRFLD**.

It's only necessary to do this the first time you run **UPO** so that the file exists.

Run the program **UPO** and you will be prompted as follows:

HIGH TO 0 Enter the first stamp you have starting with the highest value and touch **R/S.**

NEXT? Enter the remaining stamps in descending order, touching **R/S** after each, ending with zero.

When zero has been entered followed by **R/S**, the program will display **UPDATING** and the entered values will be saved.

If all you have is a sheet of 2p stamps you may need a longer display and a larger envelope. That or another trip to the Post Office.

PO

Run the program **PO** and you will be prompted as follows:

POST TOT? Enter the value of the postage required and touch **R/S.**

The results will be displayed as a string of stamp values separated by spaces such as:

12 12 2 1 …which will add up to the total that you entered at the start.

Touch **R/S** to loop back for another run with the same stamp inventory.

Run **UPO** any time your stamp inventory changes.

Program Listing follows:

Program Listing

Line	Instruction	Comments
1	**LBL "PO"**	
2	"20/2/89 V1.1"	
3	"PR"	
4	CLX	
5	SEEKPTA	Set the pointer to the beginning of the extended memory file **PR**
6	11.01001	A block of registers in main memory
7	FLSIZE	
8	1000	
9	÷	
10	+	
11	STO 00	
12	GETRX	
13	FIX 0	
14	CF 29	
15	"POST TOT?"	The value of the postage you need to make up in pence/cents
16	**PROMPT**	
17	CLA	
18	**LBL 01**	Loop that tests stamp values and subtracts a stamp that is smaller
19	**XEQ 00**	
20	ARCL X	
21	" "	
22	-	Subtract the last value from the postage total
23	X>0?	
24	**GTO 01**	
25	FIX 2	
26	SF 29	
27	**PROMPT**	The result as a series of stamp values e.g. **10 5 2 1**
28	**GTO "PO"**	Loop back for another run
29	**LBL 00**	Subroutine that tests stamp values against the postage total
30	RCL IND 00	
31	X<=Y?	If the stamp value is less that the postage value remaining add to string
32	**RTN**	
33	RDN	
34	ISG 00	
35	**GTO 00**	
36	**END**	

Line	Instruction	Comments
1	**LBL "UPO"**	
2	"20/2/89 V1.1"	
3	11.04001	
4	STO 00	
5	"HIGH TO 0"	Enter the stamps you have from highest to the lowest value, including **0**
6	**PROMPT**	
7	STO 11	
8	**LBL 01**	Loop for entering stamp values - exits when zero is entered
9	"NEXT?"	
10	**PROMPT**	
11	X=0?	Exit if zero
12	↓ **GTO 02**	
13	ISG 00	
14	STO IND 00	
15	↑ **GTO 01**	
16	**LBL 02**	
17	"UPDATING"	
18	AVIEW	
19	RCL 00	
20	INT	
21	"PR"	
22	PURFL	Deletes the old file
23	10	
24	-	
25	CRFLD	Creates a new one with the correct new size
26	LASTX	
27	+	
28	1000	
29	÷	
30	11	
31	+	
32	SAVERX	Saves the stamp inventory to the extended memory file
33	CLD	
34	**END**	