HHUC99 Programming Contest

The contest this year is to "improve" the following program by reducing the program bytes required. The technique of making the program a string and using OBJ→ to save bytes is ***NOT*** allowed. This adds so much to the run time – usually many minutes – that it is not practical. The program is a menu prompting clock adjust program to allow seconds, minutes, and hours adjustment. It displays the clock to allow the time to be visible and turns the clock off when the adjustment environment is exited.

```
'CADJ' << -40 SF { { "S+" << 8192 CLKADJ >> } { "M+" << 491520 CLKADJ >> }
          { "H+" << 29491200 CLKADJ >> } { "S-" << -8192 CLKADJ >> } { "M-"
          << -491520 CLKADJ >> } { "H-" << -29491200 CLKADJ >> } { "EXIT"
           << -40 CF 0 MENU >> } } TMENU >>
```

282.5 Bytes, # DFFFh.

Two approaches are possible. Improve the program using TMENU or use a different user interface that uses the same basic elements.

1. Turn on the clock when the program is run.
2. Provide an identified key for each of the six time adjustments.
3. Provide an "EXIT' function.
4. Turn off the clock when the program is "EXITed" or terminated.

RULES

1. The decision of the judge(s) is final.
2. The purpose of the contest is to have fun.
3. User code only, no SYSEVALs, LIBEVALs, or machine code allowed.
4. Program submittal must be by 3:00 PM Sunday August 22, 1999.
5. Transfer your program to the Judges machine. Provide a three letter initial name, your byte count, Full Name, and email or street address.
6. One submittal per person. One person per submittal.
7. By submitting a program you agree to allow it to be shared with the community.
8. You need not be present to win. Winning prizes will be shipped if you are not able to be present. It is best to have a friend accept it for you. You must tell us this when you submit your entry.
9. If a point is unclear, ***ask*** immediately. No excuses for ignorance. Any Rule changes, if something is unclear, will be officially announced during normal conference hours.
10. This contest entry instruction sheet will be available at registration Saturday Morning or with the welcome packet.
11. This is a contest between individuals, not teams.

Happy Programming.

X < > Y,

Richard Nelson

HHC 2000

Programming Contest Problem

This is a report of the results of the Programming contest conducted at HHC 2000. The rules as provided in the conference proceedings are reproduced below.

Problem Statement: Input: **A list** of ASCII numbers, 0 through 255. Output: **A list** of two strings. The first string is the contents of the input list with the ASCII numbers delimited by a space. The second string is the ASCII characters delimited by a space. There shall be no extra leading or trailing spaces in either string. See example below:

Input: { 65 66 67 68 69 70 }
Output: { "65 66 67 68 69 70" "A B C D E F" }

The winning program will be based on the product of the execution time in seconds and the byte count. HP 49 execution times made in approximate mode. Alternate list inputs of variable length using random ASCII character numbers (applied to all entries, averaged) may be used if the winner is not obvious from the example input. Either HP48 or HP49 programs accepted.

RULES

1. The decision of the judge(s) is final.
2. The purpose of the contest is to have fun.
3. At least three contestants must participate.
4. User code only, no SYSEVALs, LIBEVALs, or machine code allowed.
5. Contest program submittal must be by 3:00 PM Sunday September 10, 2000.
6. Transfer your program to the Judges machine. Provide a three letter initial name, your byte count, checksum, full name, and email or street address on the entry form.
7. This is a contest between individuals, not teams ; one submittal per person, one person per submittal.
8. By submitting a program you agree to allow it to be shared with the community.
9. You need not be present to win. The winning prize will be shipped if you are not able to be present. It is best to have a friend accept it for you. You must tell us this when you submit your entry form.
10. If a point is unclear, ***ask*** immediately. No excuses for ignorance. Any Rule changes, if something is unclear, will be officially announced during normal conference hours.
11. This contest entry instruction sheet and entry form will be available at registration Saturday Morning or with the welcome packet available upon arrival Friday.
12. You may assume machine default flag settings. Altered flag settings must be returned to default status upon program completion. Do not leave any garbage on the stack. Stack contents shall not be altered by the program except as defined by the problem statement.

Nine programs were received. Two of them didn't work correctly and one produced and error message when it was run. The qualified six programs are listed below.

'ELS' << DUP CHR 2 →LIST 1 << LIST→ 2 SWAP START " " SWAP + + NEXT >> DOLIST >>.

66.5 Bytes, # F6C3h.

'SRA' << DUP << " " SWAP + + >> → **x** << **x** STREAM SWAP CHR **x** STREAM 2 →LIST >> >>

80.5 Bytes, # 9433h.

'JGS' << DUP →STR 3 OVER SIZE 2 - SUB "" ROT + << " " SWAP CHR + + >> STREAM TAIL 2 →LIST >>

84.5 Bytes, # 6056h.

'JJS' << DUP →STR 3 OVER SIZE 2 - SUB SWAP REVLIST CHR OBJ→ "" 1 ROT START " " ROT + + NEXT TAIL 2 →LIST >>

82.0 Bytes, # F4EBh.

'BPW' << DUP →STR 3 OVER SIZE 2 - SUB SWAP 1 << CHR " " + >> DOLIST ΣLIST 1 OVER SIZE 1 - SUB 2 →LIST >>

89.5 Bytes, # CD72h.

'JDB' << C$ 22 " " SWAP ADD ΣLIST TAIL DUP2 OBJ→ ROT CHR ROT OBJ→ 2 →LIST >>

57.0 Bytes, # F78Eh

I also wrote a program and I wanted to see how I compared so I will list mine as well. It is not an entry.

'RJN' << DUP →STR 3 OVER SIZE 2 - SUB SWAP CHR LIST→ 2 SWAP START " " SWAP + + NEXT 2 →LIST >>

66.0 Bytes, # D1D0h.

Table 1 — Summary of Measurements Made on six Submitted Programs

Parameter	'ELS'	'SRA'	'JGS'	'JJS'	'BPW'	'JDB'	'RJN' Not counted
Commands	16	17	21	24	24	15	21
Check Sum	# F6C3h	# 9433h	# 6056h	# F4EBh	# CD72h	# F78Eh	# D1D0h
Bytes	66.5	80.5	84.5	82.0	89.5	57.0	66.0
6 CHR TIM	0.570_s.	0.462_s.	0.450_s.	0.678_s.	0.651_s.	1.53_s.	0.467_s.
6 CHR B*T	29.3	31.9	32.4	40.4	40.9	59.3	27.9
10 CHR B*T	43.8	49.7	49.6	59.5	61.3	71.8	42.6
26 CHR B*T	130.3	132.0	136.9	160.7	165.6	136.2	118.8
100 CHR B*T	1510	522	543	795	654	441	667
200 CHR B*T	1450	1090	1100	1990	1370	866	7030

When I announced the winner it was shortly realized that his output didn't quite meet the requirements. The winner gave the prize (HP 40G + other goodies) to Eric Smith who had the second "winning" program. Study these programs to understand some of the difficult decisions that have to be made. See the second last sentence above **RULES**. The shaded values are the ones that determined the winner. The additional data is provided for comparison. –rjn

Three programs were used to obtain the data in table 1. The first one calculates the time of execution for each program in a list, **'L1'**. The input is obtained using the list in **'D1'**. The output of **'TSTT'** is a tagged list of the program execution time in seconds rounded to three significant figures.

'TSTT' << 1 **L1** SIZE FOR **n D1 L1 n** GET RCL TIM DTAG UVAL SWAP DROP NEXT 7 →LIST -3 RND **L1** →TAG >>

84 Bytes, # E061h.

A similar program calculates a list of program bytes. This is named **'TSTB'**.

'TSTB' << 1 **L1** SIZE FOR **n D1 L1 n** GET RCL BYTES ROT ROT DROP2 NEXT 7 →LIST >>

65.5 Bytes, # 53CDh.

Both programs use generalized list processing. A program that computes the BT values is named **'TEST'**.

'TEST' << **TSTT TSTB** * -3 RND **L1** →TAG >>

40.5 Bytes, # BFA6h.

A few Thoughts

What are the programming issues posed by the contest programs?

1. What is the practical use of the program? What is it used for, and why are two strings important?

2. Looping through a string or list is very much length dependant. Understanding the command set and the limitations of each command makes the choice of commands easier. Testing the commands, to meet the requirements of the problem, is what the contest is really about. In actually there is no "best" or "winning" program except the one that meets very specifically defined parameters. See 3 below.

3. Why is there such a time difference between short and long list inputs? Table 1 only provides moderate length inputs. What are the limits of the machine?

4. How do the list processing approaches compare with the "normal;" looping approaches? Is one approach always better than the other?

5. If you participate in a contest should you have a few tools available such as the programs above to quickly evaluate your programs being considered for entry?

6. Next year there will probably be two problems, one for the HP 48, the other for the HP 49. This will make it more difficult to judge. It will certainly take longer.

HHC 2001 Programming Contest Report

Richard J. Nelson

This report consists of four major parts and includes all material related to the HHC 2001 HP48/49 Programming Contest. HHC 2001 Conference Participants decided to hold the contest on the Internet the following weekend - September 29 & 30, 2001. The contest was open to all people pre-registered for the Conference as listed on the Conference Web Page. The four parts are:

A — Problem and Rules

Normally the problem and rules are provided at Conference registration on Saturday Morning. Those who wish to participate must provide their solutions by 3PM the next day, Sunday. In this contest the Problem and rules was emailed - to those who registered as participants – on Friday afternoon at about 5 PM PDT. The contest was held just as if the participants were at the Conference in terms of resources to be used. Programs were required to be emailed and received by 5 PM PDT giving the participants an extended time to work on the problem. This was done for two reasons. The first reason is the more mathematical nature of the problem this year compared to years past. The second reason is that the problem was well documented before the contest. Providing a complete report of such an effort provides a good learning exercise for everyone in the HP User community.

Problem: Summing Integers.

Given two positive integers: **m**, **n** where **m** < **n**.
Calculate A using the following:

$$A = \frac{\Sigma E}{\Sigma O}$$

Where: A is the ratio of the Even integer sum and the Odd integer sum,
E is the sum of all even integers from **m** to **n** inclusive,
O is the sum of all odd integers from **m** to **n** inclusive.

The "best" program wins the prize and will be judged using the four sets of inputs listed in Table 1. For each input the execution time will be measured and multiplied by the number of bytes of the submitted program. This "BT" value will be averaged for the final "BT" value for the program.
Alternate BT: Alternately, the run times for each set of the four inputs may be summed and the sum multiplied by the program byte count to determine the winner. This process will be performed by a program to automate the determination of each entry's BT value.

All programs will be judged in the same way. In the event of a tie, an alternate set of inputs will be used to break the tie.

An example of a program that solves the problem is listed below.

```
'EXAMP' << DUP2 { } ROT ROT FOR n n + 2 STEP ROT 1 + ROT { } ROT ROT
        FOR n n + 2 STEP DUP HEAD IF 2 MOD THEN SWAP END ΣLIST
        SWAP ΣLIST / >>
```

119.5 Bytes, # CA91h (HP48).

Do not let this program influence you in your choice of method or choice of commands. This program is ***only*** provided as an example of a program meeting the problem requirements. Table 1 shows four values of **m** and **n** along with the value of A, Time, BT, final BT score, and alternate BT score using **'EXAMP'**

Table 1 – Typical Judging Test Data

m	n	A	Time	BT
234	298	1.03125	2.18_s.	261
244	267	0.99609375	0.593_s.	70.9
257	288	1.00367647059	0.819_s.	97.9
239	277	0.95	1.04_s.	124
			BT = 139, alternate BT = 553 (lowest BT value wins)	

Program Submittal: Complete the program submittal form which contains the following information.

1. Your Name
2. Three letter (initials) program name.
3. Your program (HP48) byte count. If you use an HP49 get this value when you transfer your program.
4. Your email address (optional if you are present).

Transfer your program to the machine used for the judging. This will be an HP48GX. Programmers will have to accept the HP48 version of their HP49 entry. Some HP49 commands place the machine in Exact mode which could be a disadvantage. Certain stack commands not available on the HP48 could provide an advantage to HP49 programs. Ask an HP48 user if in doubt. HP49 stack commands UNROT, UNPICK, PICK3, DUPDUP, NIP, and NDUPN are not valid (usable) HP48 commands for purposes of this contest.

RULES – HHC 2001 Programming Contest

1. The decision of the judge(s) is final.
2. The purpose of the contest is to have fun.
3. At least three contestants must participate.
4. User code only, no SYSEVALs, LIBEVALs, or machine code allowed.
5. Contest program submittal must be by 3:00 PM Sunday September 16, 2001.
6. Transfer your program to the Judges machine. Provide a three letter initial name, your byte count, checksum, full name, and email or street address on the entry form.
7. This is a contest between individuals, not teams; one submittal per person, one person per submittal.
8. By submitting a program you agree to allow it to be shared with the community.
9. You need not be present to win. The winning prize will be shipped if you are not able to be present. It is best to have a friend accept it for you. You must tell us this when you submit your entry form and program.
10. If a point is unclear, ***ask*** immediately. No excuses for ignorance. Any Rule changes, if something is unclear, will be officially announced during normal conference hours.
11. This contest entry instruction sheet and entry form will be available at registration Saturday Morning September 15, 2001 at 8 AM.
12. You may assume machine default flag settings. Altered flag settings must be returned to default status upon program completion. Do not leave any garbage on the stack. Stack contents shall not be altered by the program except as defined by the problem statement.
13. Use only the resources you brought with you. This does not include an Internet connection or communication with some one not attending the conference.

B — Participants And Email Correspondence During The Contest

The table below list the participants. The various forms of the information are grouped for convenient cut and paste as required. All "registered" participants submitted a program except Paul Kettler.

HHC 2001 Programming Contest Participants

1	Roger Hill	RCH	rhill@siue.edu, bpwalsh@speakeasy.net, vipond@vax2.concordia.ca, pkettler@condor.depaul.edu, vklindsay@mindspring.com, wlodek@hpcc.org, jacob.g.schwartz@lmco.com, jackie@cis.csuohio.edu, jmprange@i-is.com, zinnebe@auburn.edu, eric@brouhaha.com,
2	Brian Walsh	BPW	
3	Bill Butler	BBB	
4	Paul Kettler	P?K	
5	Vern Lindsay	VKL	
6	Wlodek Mier-Jedrzejowicz	WMJ	
7	Jake G. Schwartz	JGS	
8	Jackie Woldering	JFW	
9	James M. Prange	JMP	Roger Hill, Brian Walsh, Bill Butler, Paul Kettler, Vern Lindsay, Wlodek Mier-Jedrzejowicz, Jake G. Schwartz, Jakie Woldering, James M. Prange, & Bertram Zinner, Eric Smith
10	Bertram Zinner	ABZ	
11	Eric Smith	ELS	

1

Roger Hill #5
300 South Main St.
Edwardsville, IL 62025-2060

2

Brian Walsh
715 Braeside Pl.
Barrington, IL 60010

3

Bill Butler
3435 Westmore Ave.
Montreal Quebec
Canada H4B 1Z7

4

Paul Kettler
7435 S. Yates Blvd.
Chicago, IL 60649

5	6	7	8
Vern Lindsay #214 7470 Holbrook Lane Boise, ID 83704	Wlodek Mier-Jedrzejowicz, 42 Heathfield Road, London W3 8EJ, United Kingdom	Jacob G Schwartz 135 Saxby Terrace Cherry Hill NJ 08003	Dr. Jackie Woldering 297 East 270th Street Euclid, Ohio 44132-1601 Phone: (216) 289-2682 Email: jackie@cis.csuohio.edu
9	10	11	12
James M. Prange 7180 Springborn Rd. China, MI 48054-3601	Bertram Zinner 920 Sanders Street Auburn, AL 36830	Eric Smith #379 142 N. Milpitas Blvd. Milpitas CA 95035	

HHC 2001 Programming Contest Email Questions

In order to get everyone coordinated I sent an email to the participants "email list."

9/29/01 7:08 AM PDT

Hello Programmers,

Attached is a two page *.PDF file you should be able to read and print. Page one is the official list of participants. Page two are the rules. I will make a similar mailing after work (5 PM PDT) today with a one page attachment describing the problem.

1. Confirm ASAP that this test email was received and attachment is readable .

2. We will hold this contest as if we were at the conference. As such you have your brain, machine, and any related manuals related to your machine to use as your resource. Who can program without their AUR?

3. Correct any (missing, typically a middle initial for your program name) information on page 1.

I have no way of verifying receipt of this or the problem unless you respond.

Be sure to ask any questions. I will be at this email address most of Saturday and Sunday to answer questions. All correspondence from me for contest purposes will be to the contestant mail list on page one so everybody has the same information.

X < > Y,

Richard Nelson

All but three responded as having received and being able to open and read the *.pdf attachment. There were some questions of file and Adobe Reader version compatibility and I decided to remove all spaces in the attached file name to be safe.

9/28/2001 5:17 PDT

Hello Programmers,

Attached is the programming problem for 2001. You should act as if you are at the conference except that you will not have the distractions of the speakers :-)

Email me if you have any questions. I did not get a confirmation from three people so I hope they get this OK. I corrected several email addresses and I hope the list used here is correct. I had a lot of problems with Eudora changing addresses without me noticing it. If your name was the same as an employee Eudora changed the address to AEMF!

Good luck and email me your program typed a clearly as you can by 5 PM PDT Sunday. This is longer than if you worked on the problem at the conference. Have fun. Include your three letter initials for the program name. Include the byte count and check sum.

Email me with any questions that you may have. I will be here tomorrow at about 10 AM. I am giving my formal email signature with additional information.

Richard J. Nelson - Technical Writer
Alfred E. Mann Foundation for Medical Research
28460 Avenue Stanford, Suite 215, Valencia, CA 91355
Phone: (661) 775-3995 x3070 FAX: (661) 775-9775
Email: rjnelson@aemf.org

Questions (Saturday) #1

Hello Programmers,

See answers below. I believe that these also answer a question by Bill Butler.
Happy Programming. We are at the half way point. -RJN.

Dear Richard,
A trio of quick questions and a note:

1. Do I understand correctly that I have the full command set of the HP48GX at my disposal in the solution of the contest problem?

That is correct.

2. In the event of a tie, does the earliest submission win (in other words, is there any rush to get the program in substantially before the 5 P.M. PDT Sunday deadline?)

My worst nightmare is the situation where the optimum program is "discovered" by all the participants. I had thought that if the check sum was different the programs would be different. Not quite true. A couple of years ago a high value numeric constant was involved and few picked the same value, but otherwise the programs were similar. I spent over 30 hours of statistical measurements to try to determine a winner. It was close and "I got by". Hopefully the problem is chosen so there is lots of room for creativity and therefore many different programs. When Joseph and I held our "own" competition solving this problem we came up with - from his evaluation - identical program solutions but different code. Of course this resulted because we worked under nearly ideal circumstances - adequate time, no constraints, etc. Of course there was the "friendly competition" aspect that provided the fuel needed.

Question answer. I encourage full competition and no rush to finish to be "first" to use time to break a tie. I will use an arbitrary set of values to apply to programs that "tie". Unless they have the same code - not likely, that is what always surprises me - there is usually some timing differences that can be used.

3. I know you mentioned this somewhere earlier, but what is the prize or prizes (if there is first, second, third place, etc.) for which we are competing?

In years past we had lots of prize donations so we had three prizes. This year we have only one. I will announce the prize when I announce the winner. It will be a "nice" prize.

Thanks for the problem, and I'm warming up my calculator now. jackie

Dr. Jackie F. Woldering
Term Assistant Professor
Computer and Information Science Department
Cleveland State University
Cleveland, Ohio USA
(216) 523-7224 (Office)
(216) 289-2682 (Home)
jackie@cis.csuohio.edu

Questions (Saturday) #2

At 9/28/200109:28 PM-0500, you wrote:
Is it correct to assume that inputs are input in the order of m,n (m in stack level 2, n in stack level 1)? That is, should we assume that the lesser number is in level 2 ?

That is correct, sorry it was not clear. -rjn.

--

Brian

Questions (Saturday) #3

Hello Programmers,

See answers below.

At 9/29/200103:17 PM-0400, you wrote:

Hello Richard,

You say that you will use "an arbitrary set of values to apply to programs that tie".

If there is a tie I have failed in my choice of the problem. I hope that is not the case here.

1) With what precision will the initial timing be made? (i.e. what's a tie?)

Obviously the larger the spread for m and n the larger is the resulting sum. If I time each program using the published data and several programs are very close I will pick another set of data. I will decide on how large that set will be when I have to make that decision. Clearly you have to decide on the tradeoffs yourself. Since the "better" program performs well under a wide range of inputs that will be the criteria. You, however, have to make the tradeoffs based on the first level of testing. Blow everyone away with a short fast program with the first round and it doesn't matter what the performance is for another set of data. If, however, the competition is just as good, then you had better think of the next "level". If you knew what everyone else was doing this decision would be easy, but . . . Making these decisions is part of your strategy in preparing your program. Don't get too hung up on this. Just write the fastest shortest possible program!

2) May one at least assume that your "arbitrary set of values" will be "parity balanced" as in your "Typical Judging Test Data" example?

With the philosophy discussed above the answer is yes. The important part is to realize that all programs will be run on the same data. Cutting too many corners to gain in a small area is only one of many choices you have to make.

Bill Butler

X < > Y, Richard Nelson

Questions (Saturday) # 4

Hello Programmers,

See answer below.

Hello Richard,

I wanted to verify that the integers m and n are assumed to be *nonzero*. This may sound silly because you did say "positive integers", which strictly speaking excludes zero, but I have sometimes seen people (including physicists!) be sloppy and say "positive" when they really meant "nonnegative". No offense meant to you or Joe; I really don't expect that either of you would do this, but I just wanted to be completely absolutely unambiguously positively sure that you won't use zero in tie-breaking test data...

1 to n, m to n, but I will not use zero! -rjn

I hear a quiet roar of keys being pressed in the distance.

-- Roger

Questions (Saturday) #5

Hello Programmers,

See answer below.

At 9/29/200105:00 PM-0500, you wrote:
Hello again,

I forgot to also check on whether **LASTARG** is allowed. I guess this comes under "**default** flag settings".

Yes, and Yes. -rjn

-- Roger

Questions (Saturday) #6

Hello Programmers, See Answers below.

At 9/29/200107:11 PM-0400, you wrote:
Dear Richard,
A few questions about the program submission:

1. How can I determine the official size and checksum for the program? When using the SIZE command for the 'EXAMP' program on an HP48G, it indicates 129 bytes, not the 119.5 bytes that you show (maybe 119.5 bytes is after compression or packing?)

All program byte counts are for the program only, not the program and the name. Recall the program to the stack and execute BYTES.

2. To 'transfer the program to the judge's machine', is it sufficient to send an ASCII typed version of the program along with programmer name, three initial program name, program size and checksum, and email or street address, or do you want the program in some other form?

Since I have most of what I need on the Participants list I only need your "name" as your three initials and a typed version of the program. An arrow may be ->, <-, etc. See answer above for the check sum. Your machine should be in HEX mode and your word length should ALWAYS be set to 64.

3. If the program does not arrive on your end in runnable shape, will you notify the programmer for resubmission (if the entire submission is unreadable), or correction of a typo (assuming that it is a small correction and not an entire revision of the program), or, conversely, will you be confirming our submissions in some way?

I will key in the program and ask questions if I don't get your (the same) checksum. That should not take very long. I will confirm receipt of the correct - same check sum - program. Since email is not very reliable in its transmission time you may not hear from me until Monday morning.

4. Will the winning entry be published in HHC 2001 conference proceedings, or perhaps on a CD-ROM, or has the time for adding material passed?

The world will know the winner as soon as I know! This contest will be the most documented one ever. This Contest and the many before will be on the CD ROM. I am sure Joseph will want to "tease" the news group with the problem on Sunday evening.

May I include some discussion of my solution in the submission, or are we limited to only the requested name, byte count, etc., and program code in whatever form? The discussion cannot influence the judging of the results, which is based strictly on program performance and byte count, but it might be interesting to you and others to see how I arrived at a solution, whether mine is the optimal solution or not.

Yes! That is what the contest is about. Include all the verbiage you wish. I love to read this stuff. Joseph and I have 15 to 20 pages between us to share on this problem as well.

Thank you.
Dr. Jackie Woldering

X < > Y, Richard Nelson

Questions (Saturday) #7

Hello Programmers,

See answer below.

At 9/30/200103:55 PM-0400, you wrote:
Dear Richard,
One other thing that was not entirely clear to me. Once the program has been submitted and seen to be working correctly by you, will you accept any revisions to the program should inspiration strike before 5:00 P.M. PDT on Sunday, September 30, 2001?
jackie

I will use the program I have based on the submittal deadline. If you change your entry you must be very careful - as I must - not to get confused, etc.

This asks an important question that is addressed during the Conference. Contestant interaction, the reading of the group, etc. I am not giving any evaluation or comment on a program unless there are issues

of having the program as intended and it is the "right" one, i.e. it works, etc. I am manually keying in the programs and using the check sum to verify correctness. This works very well. These issues are better addressed with early submittals. Most programmers will have several versions to evaluate, etc. If the "wrong" one is sent and I can safely make the change I will try, but I caution you on making last minute changes. I will go out to lunch soon and I will be back to check on additional submittals.

Any comments on the nature of the problem are appreciated. It is difficult to pick a problem that is "just right" in that it offers a wide range of solutions and approaches, yet is simple enough for anyone to participate. Did you like having a sample program to resort to for testing purposes?

I have a *.PDF file of contests from 1996 to the present I can email on request after you have submitted your program.

X < > Y, Richard

Near the end of the allotted time I sent this message to all participants.

Hello Programmers,

I have six entries. There is one more hour!

X < > Y, Richard

I then sent an email announcing the 15 minute point in the Subject with no text.

After I received the entries I confirmed with the following message.

Hello Programmers,

I have a total of ten programs to compete for the prize.
The programs are:

1. JFW - 95.0 Bytes #5192h
2. ELS - 70.5 Bytes # 29B2h
3. JGS - 152.5 Bytes # 7723h
4. RCH - 67.5 Bytes # A1C6h
5. BTW - 147.0 Bytes # 457h
6. BBB - 70.0 Bytes #7EC0h
7. JMP - 85.0 Bytes # 8FA7h
8. WMJ - 95.0 Bytes # 4AFAh
9. VKL - 66.5 Bytes # 1826h
10. ABZ - 72.5 Bytes #755Eh

I include Joseph's program also. I have so many programs in my machine I am not sure which one is mine. Joseph do you have a copy? I will have to look, but not now it is not important.

11. JKH - 70.0 Bytes #CF00h

I have a program that runs the entries with all four sets of the example data and compares the entry program with the values given in Table 1. All entries get the same values except one. Three of the four cases are correct, the odd-odd case is good to 11 significant digits. Let's see how things work out.

Many thanks to you all. Keep tuned as I get data from each program. While I thought that many of you used the same (or very similar) method to get an answer the programs differed greatly. Some are very "far out". VERY NICE!

My congratulations to you all, what a collection!

Back to work, Stay tuned.

X < > Y,

Richard

C — Programs Submitted And Announcement Of The Winner.

#1. The first program was from Jackie Waldering:

Date: Sun, 30 Sep 2001 00:01:24 -0400 (EDT)
Subject: HHC 2001 Programming Contest Solution - JFW - 95 Bytes

Dear Richard,

Well, I've looked this over and have squeezed out as many bytes as I can find. I'm sending this as a text file attachment so it won't get mangled by my email sender or your reader. The last stop for it was a Linux system, so if it comes to you looking like one long line without carriage returns, just save the file, open it in MS-DOS "EDIT", and then resave the file to restore the carriage returns. Thank you for an interesting problem, and please let me know if you received everything OK, or if I need to resend the attachment. Thank you also for answering my questions so promptly; that was very helpful. Take care.
Cksm: #5192h.
jackie

P.S. Will you be making the timing program available with which you will be taking the measurements for the BT calculations? I am curious as to exactly how one would go about creating such a program, and the methods for obtaining an accurate reading at such short intervals. Thanks.

P.P.S. I was asking in an earlier email about whether or not these contest submissions might be published in conjunction with HHC 2001 not to put any time pressure on you, but rather to ask whether I might be able to get a publication credit if you should put the attachment to this email in with the HHC 2001 proceedings (every little bit helps when it comes to tenure).

Solution:

#1

```
'JFW' 95 bytes #5192h checksum
<<
  IF  OVER  DUP2  +  2  MOD  THEN
    +  1  +  DUP  2  -
  ELSE
    -  2  /  DUP  1  +
  END
  IF  ROT  2  MOD  NOT  THEN
    SWAP
  END
  /
>>
```

#2. Eric Smith

Envelope-to: rjnelson@aemf.org
Date: Sun, 30 Sep 2001 00:39:46 -0700 (PDT)
Subject: Contest entry
From: "Eric Smith" <eric@brouhaha.com>
To: <rjnelson@aemf.org>
X-Mailer: SquirrelMail (version 1.2.0 [rc1])

Richard,

Here's my entry (ELS). I might try to improve it more later today, but I might not have net access so I want to send you what I've got now just in case.

I was able to get to 78.0 bytes pretty easily, but getting it down to 70.5 took a lot of head scratching. I think it may still be possible to extract a few more bytes without completely changing the approach, but I suspect that anything a lot shorter would have to use a much different approach.

#2

```
'ELS' << SWAP 1 - 2 ->LIST 2 / DUP
FLOOR DUP 1 ADD * LIST-> DROP -
SWAP CEIL SQ LIST-> DROP - / >>

#29B2 (HP 48GX), 70.5 bytes
```

This program will not work on the 48S/SX due to the use of list processing, but the rules said it was to be judged on a GX, and didn't say that it had to run on an S/SX, so I assume this is acceptable.

The duplicated code LIST-> DROP - bothers me but so far every attempt to collapse it has taken more bytes rather than fewer.

Before I came up with the use of FLOOR and CEIL on identical copies of a list, my 78.0 byte version computed two lists whose members differed by 1, divided them by two separately, and used IP on both.

Key observations about the problem, of which I'm sure you are already quite aware:

1) An iterative approach is too darn slow. The built-in sigma function might be better, but an approach using a non-iterative formula is clearly going to be fastest.

2) Considerable savings are to be found by eliminating explicit parity (even/odd) testing of the arguments. Ideally the non-iterative formula should take care of this for you.

3) The SigmaE and SigmaO functions are obviously related to the "triangle" function: triangle(n) = (n * (n+1))/2

4) The obvious expressions for SigmaE and SigmaO will involve subtracting f(n)-f(m). Since the result will be the quotient of these expressions, it is equivalent to use f(m)-f(n).

Best regards, Eric Smith

#3. Jake Schwartz

Date: Sun, 30 Sep 2001 13:01:42 -0400
From: Jake Schwartz <jakes@magpage.com>
Subject: My Programming Contest Submission

Hi Richard - Here is my submission. I did it on the HP48, but somehow was not able to send it to the computer, due to some port problems. I have keyed it in, so hopefully an ASCII transfer will be successful from the attached file to your machine.The program file attached is called "jgs.txt". Also, I have attached an "explanation" file, with the information on how this solution was achieved. It looks like others will have long-winded explanations as well :-). That file is "Summing Integers Submission.doc" in MS Word format.

As a second backup, here is the program again:

#3

'JGS' 152.5 bytes, checksum #7723h

```
<< DUP2 DUP2 - DUP 2 – SWAP / ROT
ROT + DUP 1 – SWAP 1 + / ROT 4
ROLL
  IF 2 MOD
  THEN
    IF 2 MOD
    THEN DROP INV
    ELSE SWAP DROP INV
    END
  ELSE
    IF 2 MOD
    THEN SWAP DROP
    ELSE DROP
    END
  END
>>
```

Thanks for doing all the hard work evaluating all the solutions....

Jake

#4. Roger Hill

Date: Sun, 30 Sep 2001 13:30:42 -0500 (CDT)
Organization: Southern Illinois University at Edwardsville
Subject: Contest Entry

Hello Richard,

Okay, here's my entry. I've gotta leave right now for a concert and dinner, and won't be back till a few hours after the deadline, so this is going to have to be it. I'm sure I'll find a way to squeeze out a byte or two after it's too late... :-)

#4

```
'RCH'  << - 1 OVER ^ SWAP 2 INV +
          * SWAP -1 OVER ^ SWAP 2
          INV - * - 1 + LASTARG -
          / >>
```

67.5 bytes (excluding the label), checksum #A1C6h.

This works on an HP-48SX and presumably a 48G/GX or 49G. The execution time should be almost independent of the input integers.

Explanation: Let

S = m + (m + 1) + ... + n = sum of all integers from m to n,
T = m(-1)^m + (m + 1)(-1)^(m+1) + ... + n(-1)^n
= same sum but with (-) signs given to odd integers.

Then

SumEven = (S + T)/2, SumOdd = (S - T)/2

so that the desired result is:

A = (S + T)/(S - T) = [(S/T) + 1]/[(S/T) - 1]

(where [] means ordinary brackets, not the greatest integer function). Now it turns out that, by various summation tricks,

S = n(n + 1)/2 - m(m - 1)/2 = [(n + 1/2)^2 - (m - 1/2)^2]/2,
T = [(n + 1/2)(-1)^n + (m - 1/2)(-1)^m]/2.

(S may be fairly clear, but T is harder: I found the geometric series x^m + ... + x^n, differentiated it and multipled by x to get find mx^m + ... + nx^n, and then let x = -1). Dividing these two gives

S/T = (n + 1/2)(-1)^n - (m - 1/2)(-1)^m,

which is amazingly simple and works for all combinations of even and odd m and n!

The first two lines of the program find S/T, and the last line finds A.

NOTE: I had another entry which is shorter in byte count; it uses list operations which do not work on the 48SX but do on the 49G, and I presume on the 48G as well (I don't have a 48G to actually try it on):

'**RCH1**' << NEG 2 ->LIST -1 OVER ^ SWAP -2 INV ADD * SigmaLIST
1 - LASTARG + / >>

Only 58.5 bytes; checksum #A0B3h on the 49G (probably different on the 48G because RCH had a different checksum on the two machines). However, I compared running times for your first piece of test data, and RCH1 took about 1.6 times as long to run as RCH, while the byte reduction ratio is only about 1.15. So, while RCH1 would be more likely to win in a contest based purely on byte count, in view of the present contest's rules I'm going to let the first program RCH stand as my entry.

Roger

#5. Brian Walsh

Date: Sun, 30 Sep 2001 15:14:50 -0500
From: Brian Walsh <bpwalsh@speakeasy.net>

Subject: HHC 2001 programming contest entry - BPW

#5

```
'BPW' << { SWAP DUP DUP 2 MOD
          + OVER - NOT } { ROT +
          LASTARG - 2 / 1 - * SWAP
          } -> s t << DUP DUP 2
          MOD - s EVAL - ROT DUP
          DUP 2 MOD + s EVAL +
          t EVAL t EVAL / >> >>
Bytes: 147, Checksum: #457h
```

The stack is used, and two subroutines are stored as lists and evaluated in place.

The method used is similar to the following: To total the sum of a range of numbers, add the beginning and ending numbers together, multiply by the quantity of numbers, and divide by 2.

For example, the sum of 10 through 15 is:

10 11 12 13 14 15 <- the numbers in order

15 14 13 12 11 10 <- the numbers in reverse order
--- --- --- --- --- ---
25 25 25 25 25 25 <- the sum of each pair of numbers

25*6/2 = 75 (the sum) Brian Walsh

#6. Bill Butler

Submission

#6

```
BBB  ( #7ECOh, 70 bytes)
<< DUP ROT DUP2 - 2 MOD {  + 1
   } IFT - DUP 2 + SWAP ROT 2
   MOD { SWAP } IFT / >>
```

Remarks

The performance of this program is very close to that of another similar to it (#EAB4h, 70 bytes). Good luck with the judging!

Bill Butler

#7. James Pringe

From: <jmprange@i-is.com>
Subject: HHC Programming Contest submission
Date: Sun, 30 Sep 2001 18:55:22 -0400

Program name: JMP
Size: 85 Bytes
Checksum: # 8FA7h

#7

```
'JMP' << OVER DUP2 - 2 / ROT ROT
         + 2 / OVER FLOOR 1 +
         OVER FLOOR * SWAP CEIL
         ROT CEI L * IF ROT 2 MOD
         THEN SWAP END / >>
Size: 85 Bytes,  Checksum: # 8FA7h
```

As for how I came up with this, the problem reminded me of the exercise of adding the integers from 1 through 99 with pencil and paper. You could work out 1 plus 2 is 3 plus 3 is 6 plus 4 is 10 and so on, but it would be tedious. If you imagine the set of integers from 1 through 99, the lowest, 1, and highest, 99, average to 50; the next lowest, 2, and next highest, 98, also average to 50, and so on to 49 and 51 averaging to 50, leaving an unpaired 50. Similarly for the set of integers from 48 through 52, 48 and 52

average to 50, and 49 and 51 average to 50. Evidently you can average the lowest and highest elements and then multiply that average by the number of elements; surely a lot faster and easier than adding up the elements one at a time.

Fortunately (from the standpoint of providing a challenge and avoiding ties), our problem isn't quite that simple. To get a handle on it, I charted subsets of the set { m . . .n } using various values for m and n. For example with m=1 and n=5, the "1st subset" is { 1 3 5 } and the "2nd subset" is { 2 4 }. What I came up with is that for the 1st subset, the number of elements is '1+FLOOR((n-m)/2)' and the average value of the elements is 'FLOOR((n+m)/2)', and for the 2nd subset the number of elements is 'CEIL((n-m)/2)' and the average value of the elements is 'CEIL((n+m)/2)'. I could use the "IF 2 MOD THEN SWAP END" routine to determine which subset was even and arrange the stack accordingly for the final division. My first program ended up with the correct answer in level 1, but the value of n in level 2, thus requiring a final "SWAP DROP" to tidy it up.

Thinking that perhaps I could work with the even and odd subsets, I looked again at the charts, but didn't see any simple way to do that. I did, however, notice an alternate for the average values: for the 1st subset I could use 'CEIL((n-m+1)/2)' and for the second subset I could use 'FLOOR((n-m+1)/2)'. This did look promising (at least more elegant), so I developed another program as follows:

Checksum: # 92C3h

```
<< OVER DUP2 - 1 + 2
/ ROT ROT + 2 /
DUP2 FLOOR SWAP
CEIL * ROT FLOOR
ROT CEIL *
  IF ROT 2 MOD
  THEN SWAP
  END /
>>
```

It turns out that this is exactly the same size as my previous program, and the execution time was very nearly identical: a few ticks faster with some inputs, but a few ticks slower with other inputs. Using the values from "Table 1 - Typical Judging Test Data", I found the last program to be just a tiny bit slower on average, so I didn't submit it.

Regarding the example program and Table 1, I did use them to verify that my programs were indeed returning correct results. I could have used various values for m and n and worked out the correct result with pencil and paper, but I think that including the example program and table was a good thing.

#8. Wlodek Mier-Jedrzejowicz

Date: Mon, 1 Oct 2001 00:23:57 +0100 (BST)
Subject: HHC2001 contest entry from Wlodek as my ISP is bad
Cc: wlodek@hpcc.org

Hi Richard,

My email provider is not very quick or reliable, so I am sending this an hour before the deadline. I'll just have to hope it reaches you in time.

Well, the ISP is absolutely refusing to work, if it doesn't start up again soon then I'll try sending from my backup email account on Yahoo! Very sorry for duplication if both copies arrive!

The problem is very nice! Many approaches, easily understood, and can be done in just a few hours. Thanks!

My thoughts were that a series summation is a bad approach since m and n could be such as to have very many terms. So a formula is the better way. Then, using tests for the 4 possible combinations of even and odd could slow things down, so I preferred to use IP and CEIL.

That left a limited range of options. Given the lack of time, clever tricks with HP48G list processing and so on seemed not worth trying. So in the end here is a simple algorithm, with all calculation directly on the stack. It would probably be possible to squeeze out a step or two given time - I suspect some of your other entrants have managed it :-)

The algorithm is little more than a modification of the standard formula for the "sum of integers from 1 to n". Modify this for the sum of even integers and the sum of odd integers, then subtract the sum for m from the sum for n for the even and odd cases, divide and there you are Or, at least, here I am, hoping email will work
:-)

Anyway, here is my entry,

#8

```
'WMJ'  << DUP2 2 / IP SQ LASTARG
          + SWAP 1 - 2 / IP SQ
          LASTARG + - ROT ROT
          1  + 2 / IP SQ SWAP 1 - 2
           / CEIL SQ -  /  >>
WMJ  95 bytes  # 4AFAh
```

Best Wishes,

SWAP see above :-) Wlodek

#9. Vern Lindsay

#9

```
'VKL'  << SWAP OVER R->C 0 DUP
          ROT C->R FOR i i + SWAP
          NEXT IF ROT 2 MOD THEN
          SWAP END
          / >>
68.5 Bytes,  Check sum  EF1Eh
```

#10. Bertram Zinner

Name: Bertram Zinner
I am number 10
Program Name: ABZ

#10

```
'ABZ' << DUP 2 + 2 MOD DUP DUP
         NOT - ROT * 1 - + +
         LASTARG DROP 2 MOD
         DUP NOT - + LASTARG -
         / >>
Bytes 72.5 #755Eh
```

Announcement Of The Winner

HHC 2001 Programming Contest Results

Ten entries were received with the programs confirmed with the author as being correct based on the check sum. I used the same data as shown in Figure 1 calculating the BT value. The table below shows my measurements. Since the results are not very close, a single winner is obvious. I didn't test any further. The table below has had clarification notes added to the original one sent. Decimal point errors were also corrected. The last column under the Execution time heading was changed to Sum.

Congratulations Go To Bill Butler

The prize is a $50.00 Gift card from Radio Shack. Will that work OK Bill? Let me know.

#	Program	Bytes	C. Sum	Execution Time (using TIM) in seconds					BT (2)	D% (3)
				234/298	244/267	257/288	239/277	Sum (1)		
1	JFW	95.0	5192h	0.02764	0.02761	0.02732	0.02705	0.1096	10.47	67
2	ELS	70.5	29B2h	0.5103	0.5102	0.5104	0.5098	2.041	143.9	2188
3	JGS	152.5	7723h	0.03810	0.03924	0.04319	0.04338 (4)	0.1639	25.00	298
4	RCH	67.5	A1C6h	0.03623	0.03662	0.03704	0.03641	0.1463	9.875	57
5	BTW	147.0	457h	0.07661	0.07694	0.7754	0.07724	0.3083	45.32	621
6	BBB	70.0	7EC0h	0.01950	0.02516	0.02376	0.02141	0.08983	6.288	REF
7	JMP	85.0	8FA7h	0.03702	0.03825	0.03917	0.03805	0.1525	12.96	106
8	WMJ	95.0	4AFAh	0.05251	0.05225	0.05254	0.05210	0.2094	19.89	216
9	VJL	66.5	1826h	0.4124	0.1659	0.2138	0.2562	1.048	69.71	1008
10	ABZ	72.5	775Eh	0.03242	0.03282	0.03322	0.03251	0.1310	9.495	51
X	JKH	70.0	CF00h	0.02726	0.02728	0.02932	0.02573	0.1096	7.671	22

Notes: (1) Sum of the times for the four cases of odd/even m,n combinations.
(2) Sum column multiplied by the Bytes column. Lowest value in this column wins.
(3) Percentage increase over the winner's value. Difference is far greater than measurement error.
(4) The old double division error "snuck" into Jakes program causing a 11D accuracy. This was

judged as acceptable, especially since he investigated and reported it.

I will gather all the programs into one document and get it off to everyone tomorrow. Feedback welcome. We have lots of explanations that are well worth reading. These will also be gathered into a single document.

Happy Programming,

Richard J. Nelson
September 30, 2001

NOTE: This report was not finished in one day. A crash project at work consumed a week.

D — Problem Discussions

This problem was especially interesting because of the different approaches that were used for its analysis. While it was mathematical in nature, the problem itself was simple. Most participants included notes with their program. Several, however, spent the time to explain their approach in greater detail. Four such documents are included here including one I wrote for a handout in 1998 as part of an HP48/49 programming class conducted by Joseph Horn and myself. Take the time to study each document, it is very educational.

Since this problem is so well documented I am wondering if someone will distill the best of each and wrote an even better program. One of the goals of such exercises is to arrive at the "best" program. Of course this may be objectively defined as the lowest BT value, but there are other criteria for "best" as well. Some of these I think about in selecting the problem to insure that there is a a a "fall back" means of selecting the best program if two programs have the same BT value. One thought was meeting the requirements of the equation for solving for A. Most programs ended with the division to calculate the decimal value of the ratio. Jake took a different approach and there is no single ending divide in his solution. Is a program that actually provides the sum of the Evens and the sum of the Odds "better" than one that doesn't? For example, if the sums are –12 and –18 (m=1, n=5) the division produces the correct decimal answer, but the Even and Odd sums (6 and 9) are not calculated. If two programs had the same BT score would it be proper to award the prize to the program that actually calculated the sums? By the way four of the ten programs did calculate the two sums – VKL, WMJ, JMP, and ELS. Not counting JGS the other five programs calculated values that had the same ratio. If you use m=1, and n=5 as inputs these programs produce the following: ABZ = -4 & -6, BBB = 4 & 6, BPW = -12 & -18, RCG = -4 & -6, and JFW = 2 & 3. Note that JKH = 2 & 3. The latter two programs have the fraction in reduced terms.

VKL used R? C and C? R in his program. This was unexpected and I had difficulty getting his check sum. Flag settings were checked, etc. but the problem turned out to be my error in key in the program.

One question was asked regarding zero. I have not checked the programs for this issue. Which ones fail if an input is zero? All of them? If m=0 and n=5, what should the answer be? Should it be the same as m=1?

Would the problem be significantly different if negative inputs were allowed? I decided that nothing would be gained by doing this always favoring the most simple problem. I only wish that we were able to document all of our past problems as well. *Richard J. Nelson, September 8, 2001.*

HPCC 2002 Programming Contest

Richard J. Nelson

Problem Definition

Making The Best Cut. You need to cut business cards, which are 3.5" long by 2" wide from a sheet of paper stock 11" long by 8.5" wide. Using consistent units, and the convention for the paper stock dimensions as L x W, and the desired cut size as a x b, write a program that has inputs and outputs as follows.

Input		Output	
4: W	*Note: W & L may be in any order.*		
3: L		3: % Loss: nn.n	*Note: A tagged real.*
2: a	*Note: a & b may be in any order.*	2: L: a or b	*Note: A tagged real.*
1: b		1: N (No. of pieces)	*Note: A real.*

The inputs of the paper stock size are made first in any order, i.e. W ↑ L or L ↑ W. The desired cut size dimensions are entered next, also in any order, i.e. a ↑ b or b ↑ a. After the four reals are entered the program is run and three output values are calculated.

The first program output is the percentage of scrap, or loss, of the paper stock material. This value is displayed on level three with the tag "% Loss" **to one (rounded) decimal place**. Note space between % and "L".

The second program output value tells you how to cut the paper stock by dividing the paper stock length, L, by the cut dimension a or b. The paper stock dimension L is the longest dimension.

The third program output is the maximum number of pieces that may be cut from the paper stock. Figure 1 shows an example 1 of table 1 of the best cut.

Obviously there are two possibilities of cutting the paper stock. L is cut by "a" length or L is cut by "b" length. The case where the paper stock is square is undefined. The output values for example 1 of table 1 are:

3: % Loss: 10.2

2: L: 3.5

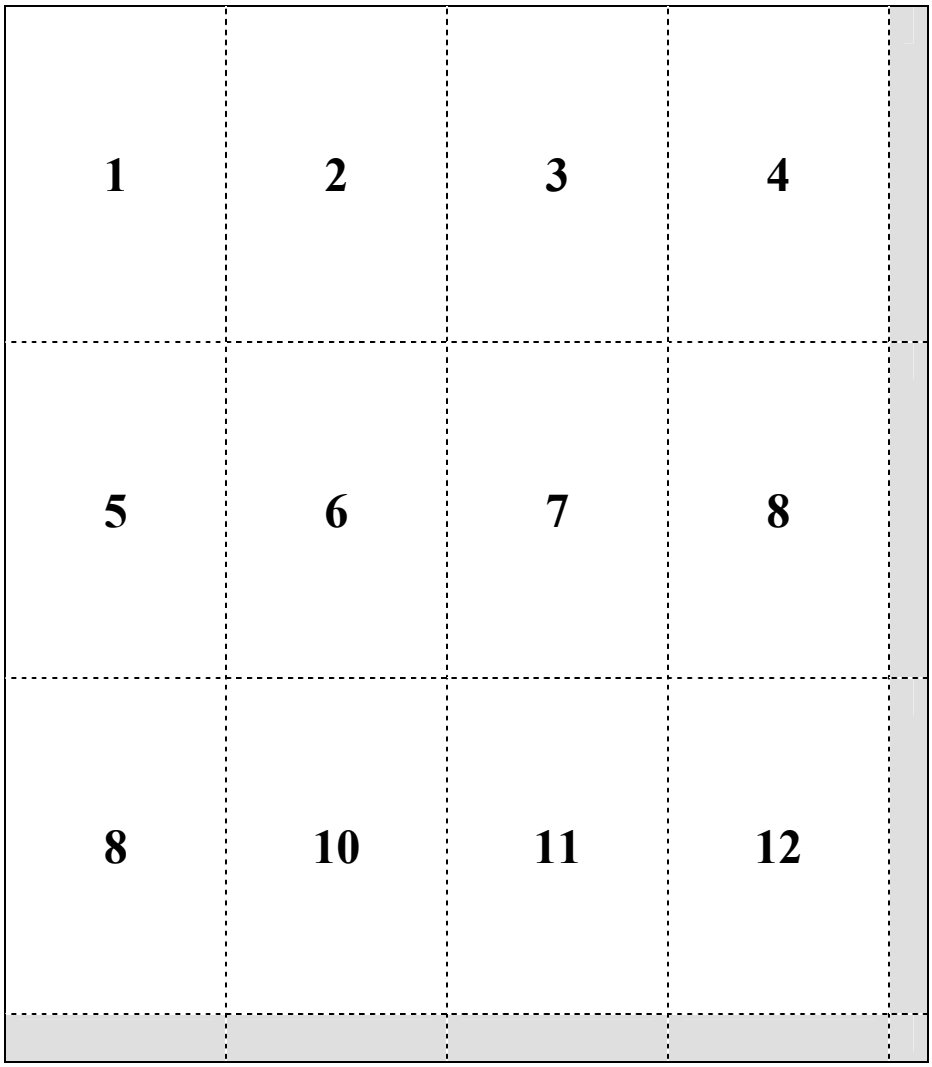
1: 12

Table 1 has data for several problems. The winning program will have the fewest bytes and run in the shortest time as determined by the product of the bytes and the seconds. See the rules below.

Table 1 — Typical Problem Data

	Inputs				Outputs			Comments
Ex	L	W	a	b	% Loss:	L:	Pieces	
1.	11	8.5	3.5	2	10.2	3.5	12	U.S. Business Card
2.	11.69	8.27	3.375	2.125	25.8	2.125	10	U.K. Business Card
3.	14	8.5	3.25	1.25	16.8	3.25	24	Labels

Figure 1 — Example 1. Typical US business cards cut from standard US Paper. Shaded area represents the scrap. Zero loss is assumed for the cuts.



Program Submittal

Complete the program submittal form or use a piece of paper, which contains the following information.

1. Your Name
2. Three letter (your initials) program name.
3. Your program (HP48) byte count. If you use an HP49 get this value when you transfer your program.
4. Your program (HP48) check sum in Hex.
5. Your email or street address.

Transfer your program to the machine used for the judging. This will be an HP48GX. HP49 programmers will have to accept the HP48 version of their HP49 entry. Some HP49 commands place the machine in Exact mode which could be a disadvantage. Certain stack commands not available on the HP48 could provide an advantage to HP49 programs. Ask an HP48 user if in doubt. HP49 stack commands UNROT, UNPICK, PICK3, DUPDUP, NIP, and NDUPN are not usable HP48 commands for purposes of this contest. See rule #14 which defines the winning criteria.

Rules

1. The decision of the judge(s) is final.
2. The purpose of the contest is to have fun.
3. At least four contestants must participate.
4. User code only, no SYSEVALs, LIBEVALs, or machine code allowed.
5. Contest program submittal must be by 3:00 PM Sunday September 22, 2002.
6. Transfer your program to the Judges' machine. Provide a three letter initial name, your byte count, HP48 checksum, full name, and email or street address on the entry form.
7. This is a contest between individuals, not teams; one submittal per person, one person per submittal.
8. By submitting a program you agree to allow it to be shared with the community.
9. You need not be present to win. The winning prize will be shipped if you are not able to be present. It is best to have a friend accept it for you. You must tell us this when you submit your entry form and program.
10. If a point is unclear, ***ask immediately***. No excuses are accepted for ignorance or unresolved issues. Any Rule changes, if something is unclear, will be officially announced during normal conference hours.
11. This contest entry instruction sheet and entry form will be available at registration Saturday Morning September 21, 2002 at 9 AM and at the Friday evening get-together at 7:30 PM.
12. You may assume machine default flag settings. Altered flag settings must be returned to default status upon program completion. Do not leave any garbage on the stack. Stack contents shall not be altered by the program except as defined by the problem statement.
13. Use only the resources you brought with you. This does not include an Internet connection or communication about the problem with anyone else.
14. Program Submittal must be a single stand-alone program object not a directory or collection of programs. No calls to other programs or external libraries allowed.

15. The winning program will meet the problem statement and have the lowest product of the byte count and the run time. The Hackers ROM TIM program will be used to determine the run time.

HP48/49 Programming Contest

September 20 & 21 Radisson Hotel, Newport Beach California

GOAL: Write a program that removes "forbidden" characters from a string.

SPECIFICS: The level-two input is the original string, randomly containing zero or more forbidden characters. The level-one input is a string containing the forbidden characters to be removed from the original string. Example:

Input:

L2: "ABC?DE#F#" <-- the original string
L1: "#?" <-- the forbidden characters

Output:

L1: "ABCDEF"

The winning program will be based on the lowest product of the execution time in seconds and the byte count. HP 49 execution times are best made in approximate mode. Alternate stringt inputs of variable length using random ASCII characters (applied to all entries) may be used if the winner is not obvious from the example input. Either HP48 or HP49 written programs are accepted but all programs must run on an HP48GX.

RULES

1. The decision of the judge(s) is final.
2. The purpose of the contest is to have fun.
3. At least four contestants must participate.
4. User code only, no SYSEVALs, LIBEVALs, or machine code allowed.
5. Contest program submittal must be completed by afternoon break on Sunday September 21, 2003.
6. Transfer your program to the Judges machine. Provide a three letter initial name, your byte count, checksum, full name, and email or street address on the entry form.
7. This is a contest between individuals, not teams; one submittal per person, one person per submittal.
8. By submitting a program, you agree to allow it to be shared with the community.
9. You must to be present to win.
10. If a point is unclear, ***ask*** immediately. No excuses for ignorance. Any Rule changes, if something is unclear, will be officially announced during normal conference hours.
11. This contest entry instruction sheet and entry form will be available at registration Saturday Morning.
12. You may assume machine default flag settings. Altered flag settings must be returned to default status upon program completion. Do not leave any garbage on the stack. Stack contents shall not be altered by the program except as defined by the problem statement.
13. Work alone. Do not consult the Internet. The AUR, Owner's Manual, or other books are permissible.

Happy Programming. Richard J. Nelson

Report on the HP49G Programming Contest at HHC 2003

by Joe Horn

There were two Programming Contests at HHC 2003, one for the HP48, and one for the HP49. The HP48 contest was written by Richard Nelson. I modified Richard's contest slightly to create the HP49 contest. The following is a report about the latter, the entries to it, and the winner.

----- THE CONTEST RULES ----------

Task: Write a program that generates a random *integer* of any specified length, with no leading zeros.

To qualify, the program must actually produce integers of exactly the desired length, *and* they must be sufficiently random to convince the judge of their randomness.

The winner will be the fastest (tested over a range of inputs). Program size will not be considered. All other rules for HHC Programming Contests apply (e.g. 100% User RPL, a single stand-alone program object, etc.)

----- THE QUALIFICATION TESTS ----------

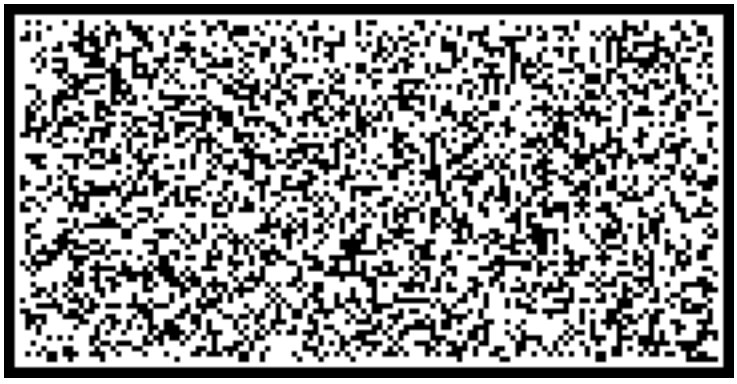
Two automated tests qualified the entries:

Test #1: Inputs of 1, 2, 11, 12, 13, and 1000 were tested for output length. Any failure disqualified the entry. Fishy-looking outputs prompted re-testing until all suspicion was removed.

Test #2: Each entry was forced to generate 4000 pairs of integers which were then plotted as single pixels on the screen. This is a powerful technique for testing randomness, since the human eyeball is adept at recognizing non-random patterns. Entries that generated patterned graphs were disqualified.

Note: I needed to be very generous regarding Test #2, because if I got too picky then nobody would have qualified and the contest would have been cancelled and that would have spoiled all the fun. ;-)

----- THE ENTRIES ----------



DMS

DMS was disqualified for not handling leading zeros, as can be seen clearly in its graph above. When leading zeros are not allowed, the bottom 10% and left 10% of the graph are blank.

CY2

CY2 has quite a bit of visible non-randomness, but it was good enough to qualify.

WMJ

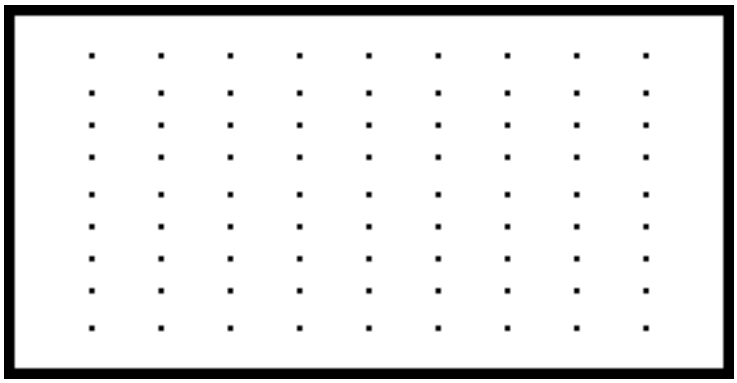
WMJ qualified as convincingly random.

TJS

TJS qualified as convincingly random.

RCH

RCH was disqualified for two reasons, both obvious in the graph above: it failed to disallow leading zeros, and it always had a zero as the last digit.



JGS

JGS was disqualified for being insufficiently random, to put it mildly.

TTTS2

TTTS2 handled leading zeros by replacing them with ones, as can be plainly seen in the dense bands from 10% to 20% from the bottom and left edges of the graph above. It was sufficiently random to qualify, but it was written in Assembly Language, and was therefore disqualified.

BZ

Similar to TTTS2, BZ replaced leading zeros with ones, but qualified anyhow.

THE WINNER ----------

CY2, by Cyrille de Brebisson. Cyrille used the clever technique of PEEKing a random location in ROM and converting it to an integer using 100% User RPL! Although the program is ugly, it's fast!

The program listings will be added to this report as soon as the Conn4x program is operational.

User RPL Programming Contest

Sept 25-26, 2004 / Radisson Hotel, San Jose, California

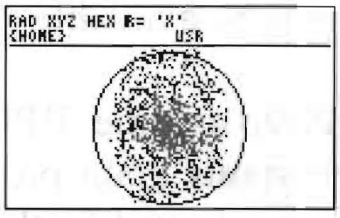
Goal:

Write a program that graphs a circle and then fills it with random dots.

Specifics:

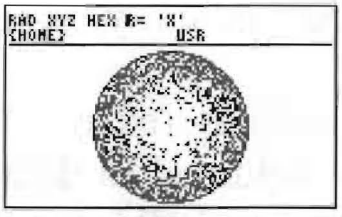
The program must graph a large circle (at least 60 pixels high for all HP48 & HP49 models), then plot random pixels inside that circle until approximately half the pixels are dark, then exit without user intervention, leaving the graph displayed. The dots must be random, that is, (a) they must be different each time the program is run, and (b) they must LOOK random (patternless) to the judge. The entire graphing process must be visible as it runs. No dots allowed outside the circle. The distribution of the pixels inside the circle must not be dense near the center, nor near the circle, nor anywhere else, but not patterned either. A violation of any of the above disqualifies the program.

Examples:



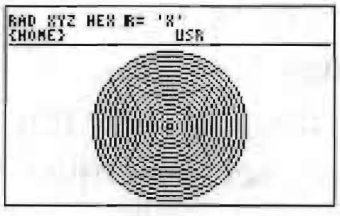
Bad.

Thumbprint?



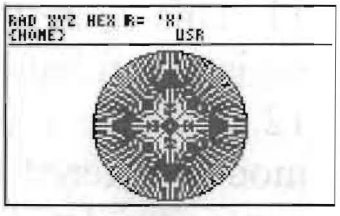
Bad.

Death star?



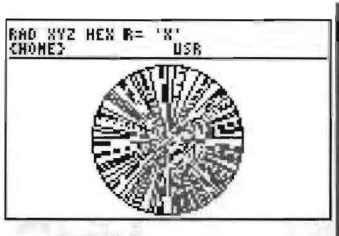
Bad.

Oriental rug?



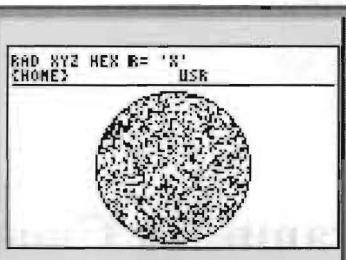
Bad.

TV test pattern?

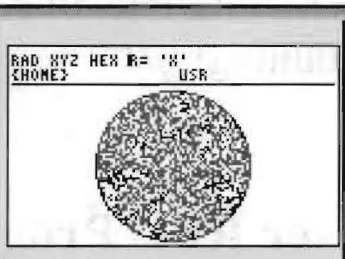


Bad.

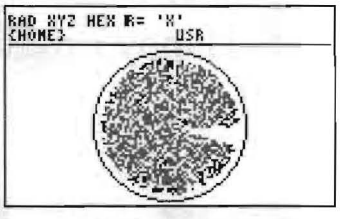
Dirty fan?



Good.



Good.



Bad.

Pizza or Pac-man?

Winning:

The winner will be the smallest program (in bytes).

Rules:

1. The decision of the judge is final.
2. The purpose of the contest is to have fun.
3. At least two contestants must participate.
4. Plain-vanilla User RPL only; no SYSEVALs or other monkey business.
5. Contest program submittal must be completed by afternoon break on Sunday September 26, 2004.
6. Include your initials in the name of your program. Transfer your program to the judge's machine. Make sure your checksum is the same as the judge's.
7. This is a contest between *individuals, not teams*; one submittal per person, one person per submittal.
8. By submitting a program, you agree to allow it to be shared with the community.
9. You must to be present to win.
10. If a point is unclear, ***ask*** immediately. No excuses for ignorance. Clarifications will be officially announced during conference hours.
11. This contest entry instruction sheet and entry form will be available at registration Saturday Morning.
12. Assume machine default flag settings (except HP49; assume RPL mode). Altered flag settings must be returned to default status upon program completion. Stack contents before and after the program must be the same.
14. *Work alone*. Do not consult the Internet. The AUR, Owner's Manual, or other books are permissible, of course.
15. Happy Programming! *~jkh~*