HappyCalc Manual

2012-11-15

Eiiti Wada (IIJ Research Laboratory)
eiiti.wada@nifty.com

Happy Hacking Calculator

Happy Hacking Calculator (HappyCalc) is a pocket calculator for integer arithmetics in the reverse Polish notations (RPN), implemented for both iPod touch and iPhone. The elements of the stack for processing the expressions are *2's complement numbers of 64 bits*. The *depth of the stack is 16*; when the numbers are pushed in more than 16 times, the lowest element of the stack will be lost. 0's are popped up from the bottom of the stack.

The HappyCalc has the *basic states* with 'Input' and 'Ready', and the *radix states* with 'Hex', 'Dec' and 'Oct'.

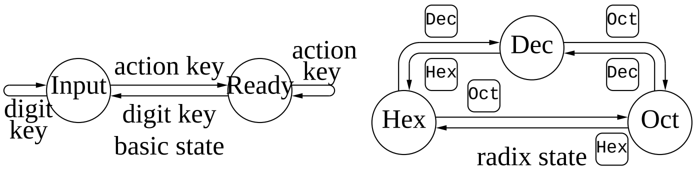
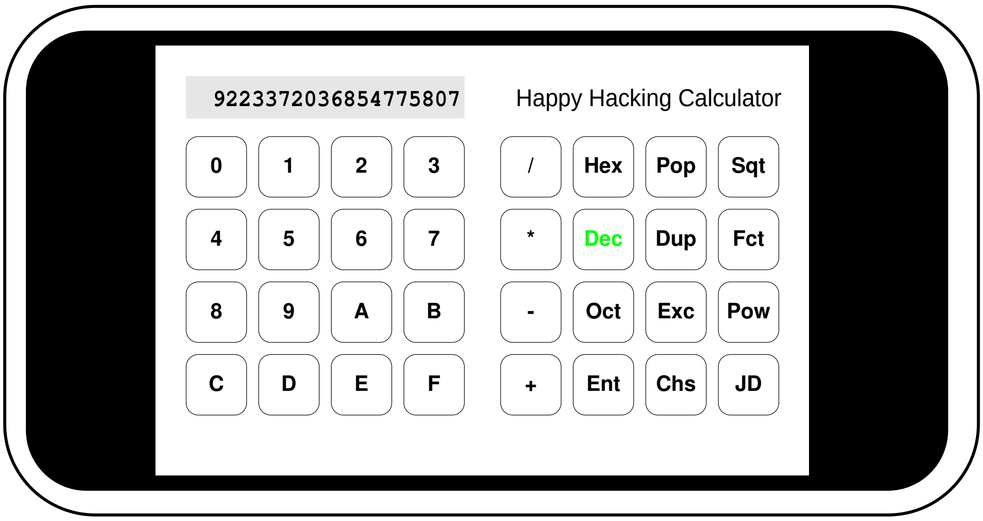


Figure 0 State transitions of the HappyCalc

The HappyCalc has 16 *digit keys* of `0`, `1`, ···, `F` on the left and 16 *action keys* of `/`, `*`, ···, `JD` on the right. A small window above the digit keys is for displaying the results.


9223372036854775807

Happy Hacking Calculator

Figure 1 The interface of the HappyCalc

The radix states are used for the radix conversion of input/output numbers and binary numbers inside of HappyCalc. Radix state can be changed with the action keys `Hex`, `Dec` and `Oct`, which set the state to 16, 10 and 8 respectively. The state is remembered in *ir*.

The present radix state is shown with the green letter. In the figure above, Dec is shown in green. When the HappyCalc is switched on, the radix is set to 'Dec'.

When any digit key is pressed in the 'Ready' state, the basic state goes to 'Input', the stack is pushed in and a sequence of digits until next action key is placed on the new stack top. Digit keys `0` to `9` input integers of 0 to 9, and `A` to `F` input 10 to 15 regardless the radix state.

Action keys perform operation on the operands on the top of the stack and replace the stack top with the result, which is displayed in the window. The basic state moves to 'Ready'.

Here, $s[ix]$ indicates the stack top; $s[ix-1]$ is the content in the stack immediately below the top. ix is the index of the stack top, and ir is the radix register. (ix is increased or decreased in modulo 16.)

Note: No backspace key is provided. Incorrect input value must be discarded with Pop.

Note: Almost all operations end instantly. Factorization of primes larger than 2^{45} will take time. Factorization of $2^{47}-115$ needs 7 seconds. During the operation, Fct key remains in blue.

Operations

+ Addition. $s[ix-1] \leftarrow s[ix-1]+s[ix]$, $s[ix] \leftarrow 0$, $ix \leftarrow ix-1$.

- Subtraction. $s[ix-1] \leftarrow s[ix-1]-s[ix]$, $s[ix] \leftarrow 0$, $ix \leftarrow ix-1$.

* Multiplication. $s[ix-1] \leftarrow s[ix-1]\times s[ix]$, $s[ix] \leftarrow 0$, $ix \leftarrow ix-1$.

/ Division. $n \leftarrow s[ix-1]$, $d \leftarrow s[ix]$. If $d \neq 0$, $r \leftarrow n\%d$, $q \leftarrow (n-r)/d$, (When $0<d$, $0 \leq r < d$; when $d<0$, $d \leq r < 0$.) $s[ix] \leftarrow q$, $s[ix-1] \leftarrow r$. If $d=0$, exception*.

Hex Hexadecimal. $ir \leftarrow 16$.

Dec Decimal. $ir \leftarrow 10$.

Oct Octal. $ir \leftarrow 8$.

Ent No operation.

Pop Delete. $s[ix] \leftarrow 0$, $ix \leftarrow ix-1$.

Dup Duplicate. $s[ix+1] \leftarrow s[ix]$, $ix \leftarrow ix+1$.

Exc Exchange. $t \leftarrow s[ix]$, $s[ix] \leftarrow s[ix-1]$, $s[ix-1] \leftarrow t$.

Chs Change Sign. $s[ix] \leftarrow -s[ix]$.

Sqt Square Root. $n \leftarrow s[ix]$. If $n \geq 0$, $q \leftarrow \lfloor\sqrt{n}\rfloor$, $r \leftarrow n-q\times q$, $s[ix+1] \leftarrow q$, $s[ix] \leftarrow r$, $ix \leftarrow ix+1$. If $n<0$, exception*.

Fct Factorization. $n \leftarrow s[ix]$. If $n \geq 2$, $r \leftarrow n/p^k$, $k \geq 1$ (where $p \geq 2$ is the smallest factor of n; p is not a factor of r.) $s[ix+2] \leftarrow p$, $s[ix+1] \leftarrow k$, $s[ix] \leftarrow r$, $ix \leftarrow ix+2$. If $n<2$, exception*.

Pow Power. $b \leftarrow s[ix-1]$, $p \leftarrow s[ix]$. If $p>0$ or ($p=0$ and $b \neq 0$), $s[ix-1] \leftarrow b^p$, $s[ix] \leftarrow 0$, $ix \leftarrow ix-1$. If $p<0$ or ($p=0$ and $b=0$), exception*.

JD Julian Day. $s[ix] \leftarrow$ Julian day of noon of year y, month m, day d, represented as $y\times 10000+m\times 100+d$ in $s[ix]$.

* In case of exception, the name of action key turns to red, and the stack is unchanged..

Examples

Stack

Relations of input and stack are shown in Figure 2. (a) Input `111`. Stack top is 111. (b) When two numbers are input without operation in between, press Ent. (c) Since it becomes ready for input next number, input 222, which is placed above 111. (d) Press Ent for the next number. (e) Input 333 to be placed above 222. (f) Press +. 2 operands on the stack top are added. (g) Press +. 666 is placed on the stack top. (h) Input 777. (i) Press Exc. 2 numbers on the stack top are exchanged. (j) Press -. The result of subtraction remains on the stack. (k) Press Chs. Sign is reversed. (l) Press Dup. The number of the stack top is copied. (m) Press *. Multiplication. (n) Input 110. (o) Press /. Quotient 112, remainder 1 are kept on the stack. (p) The quotient is seen in the window, and Pop will show the remainder. The result of these operations are the same in any radix.

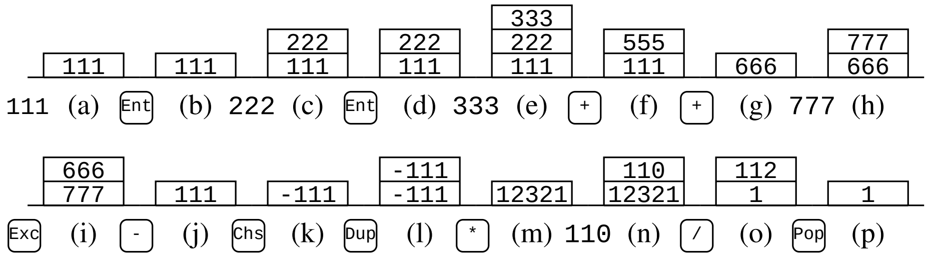

111
111
222
111
222
111
333
222
111
555
111
666
777
666
111 (a) Ent (b) 222 (c) Ent (d) 333 (e) + (f) + (g) 777 (h)
666
777
111
-111
-111
-111
12321
110
12321
112
1
1
Exc (i) - (j) Chs (k) Dup (l) * (m) 110 (n) / (o) Pop (p)

Figure 2 Stack transition (Arithemtic)

Radix change

2 Ent 10 Pow 1 - (→ 1023) Hex (→ 3FF) Oct (→ 1777). During number input, radix may not be changed.

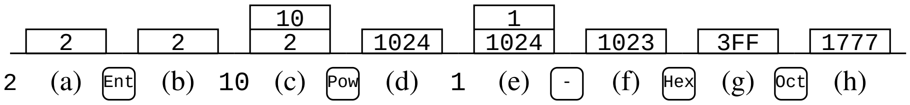


Figure 3 Stack transition (Radix)

Quotient and remainder

Sign of remainder is that of divisor. Be careful when the divisor is negative.

100 Ent 8 / (→ quotient 12 remainder 4),	200 Ent 8 / (→ quotient 25 remainder 0),
100 Chs 8 / (→ quotient −13 remainder 4),	200 Chs 8 / (→ quotient −25 remainder 0),
100 Ent 8 Chs / (→ quotient −13 remainder −4),	200 Ent 8 Chs / (→ quotient −26 remainder −8),
100 Chs 8 Chs / (→ quotient 12 remainder −4),	200 Chs 8 Chs / (→ quotient 24 remainder −8).

Square root of $2^{63}-1$

Hex 7FFFFFFFFFFFFFFF (15 F's) Sqt (→ B504F333, remainder 1615E23D6) Dup * (→ 7FFFFFFE9EA1DC29) + (→ 7FFFFFFFFFFFFFFF, the original number).

Square root of 2

Dec 20000000000000000(16 0's) Sqt (→ 141421356 = $\lfloor\sqrt{2}\times 10^8\rfloor$) Dup * (→ 19999999932878736) + (→ 20000000000000000).

Factorization of $2^{63}-1$

$2^{63}-1 = 7\times 7\times 73\times 127\times 337\times 92737\times 649657$.

Dec 2 Ent 63 Pow 1 - (→ 9223372036854775807) $($ Fct Pop Pop $)^6$ (→ 7, 2, 188232082384791343, 73, 1, 2578521676503991, 127, 1, 20303320287433, 337, 1, 60247241209, 92737, 1, 649657, 649657, 1, 1).

Factorization of 20!

`Dec` 2 `Ent` 3 `*` 4 `*` ... `*` 9 `*` A `*` B `*` ... `*` F `*` 16 `*` 17 `*` 18 `*` 19 `*` 1A `*` (→ 2432902008176640000 = 20!).

$20! = 2^1 \times 3^1 \times 2^2 \times 5^1 \times (2^1 \times 3^1) \times 7^1 \times 2^3 \times 3^2 \times (2^1 \times 5^1) \times 11^1 \times (2^2 \times 3^1) \times 13^1 \times (2^1 \times 7^1) \times (3^1 \times 5^1) \times 2^4 \times 17^1 \times (2^1 \times 3^2) \times 19^1 \times (2^2 \times 5^1) = 2^{18} \times 3^8 \times 5^4 \times 7^2 \times 11^1 \times 13^1 \times 17^1 \times 19^1$ Here, `Fct` (→ 2 = smallest factor) `Pop` (→ 18 = power) `Pop` (→ 9280784638125) `Fct` (→ 3 = smallest factor) `Pop` (→ 8 = power) `Pop` ... .

Julian astronomical day

Julian day of October 20, 2008:

20081020 `JD` (→ 2454760)1 `+` 7 `/` `Pop` (→ 1 = Monday. (Day of week is given by (JD + 1)%7).

Julian day of January 1, −4712:

4712 `Chs` 10000 `*` 101 `+` `JD` (→ 0. Note: −47119899 `Ent` 10000 `/` (→ quotient −4712 remainder 101)).

The life of the Bard

William Shakespeare was born on April 23, 1564 and died on April 23, 1616. How many days did he live?

16160424 `JD` 15640423 `JD` `-` (→ 18984 days).

Repeated digits

`Dec` 12345679012345679 (omitting 8's) `Ent` 81 `*` (→ 999999999999999999) `Sqt` (→ 999999999)

`Hex` 123456789ABCDF (omitting E)'s `Ent` E1 `*` (→ FFFFFFFFFFFFFF)

`Oct` 1234570123457012345 7 (omitting 6's) `Ent` 61 `*` (→ 777777777777777777777)

The case of twice 1...7 `Oct` 1234570123457 `Ent` 61 `*` (→ 77777777777777 even 7's) `Sqt` (→ 7777777).

Year 2038

UNIX time clock will overflow on 2^{31} seconds after midnight (0h) of January 1, 1970. When does the overflow occur?

2 `Ent` 31 `Pow` (→ 2147483648) 86400 `/` (→ 24855 days later) 365 `/` (→ 68 years later = 2038) 4 `/` (→ 17 leap years) `Exc` `Pop` (→ 17, subtract 17 from 35 in the stack) `-` (→ 18. (January) 18 will count 24855 days. Calculate time of January 19) `Pop` (→ 11648 seconds) 3600 `/` (→ 3 o'clock) `Pop` (→ 848) 60 `/` (→ 14 minutes) `Pop` (→ 8 seconds. Thus January 19, 2038 3h 14m 8s).

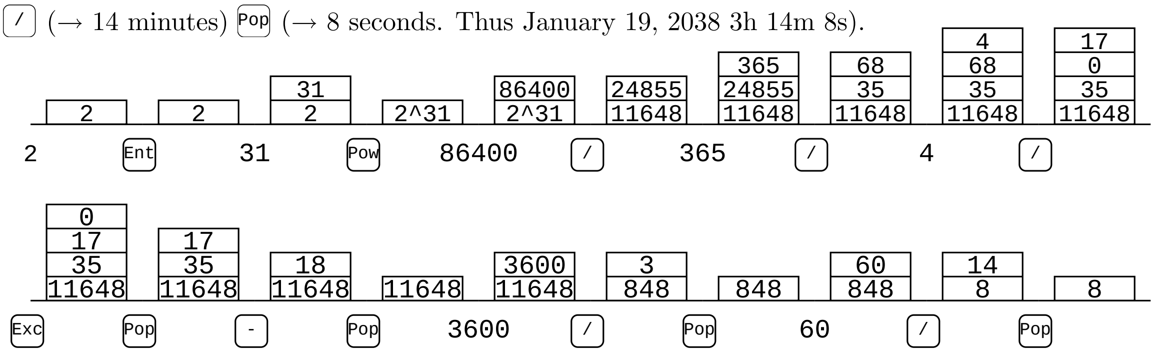


Figure 4 Unix time overflow

Continued fraction of π

3141592654 `Ent` 10 `Ent` 9 `Pow` `/` (→ 3) `Pop` (→141592654) 10 `Ent` 18 `Pow` `Exc` `/` 10 `Ent` 9 `Pow` `/` (→ 7) `Pop` (→ 62513285) 10 `Ent` 18 `Pow` `Exc` `/` 10 `Ent` 9 `Pow` `/` (→ 15) `Pop` (→ 99699762) 10 `Ent` 18 `Pow` `Exc` `/` 10 `Ent` 9 `Pow` `/` (→ 1) `Pop` (→ 3411839) 10 `Ent` 18 `Pow` `Exc` `/` 10 `Ent` 9 `Pow` `/` (→ 293) `Pop` (→ 97065834).

$\pi = 3 + /\!\!/ 7, 15, 1, 293, ... /\!\!/ \approx 355/113$. The correct value of the 4th term is 292.

end