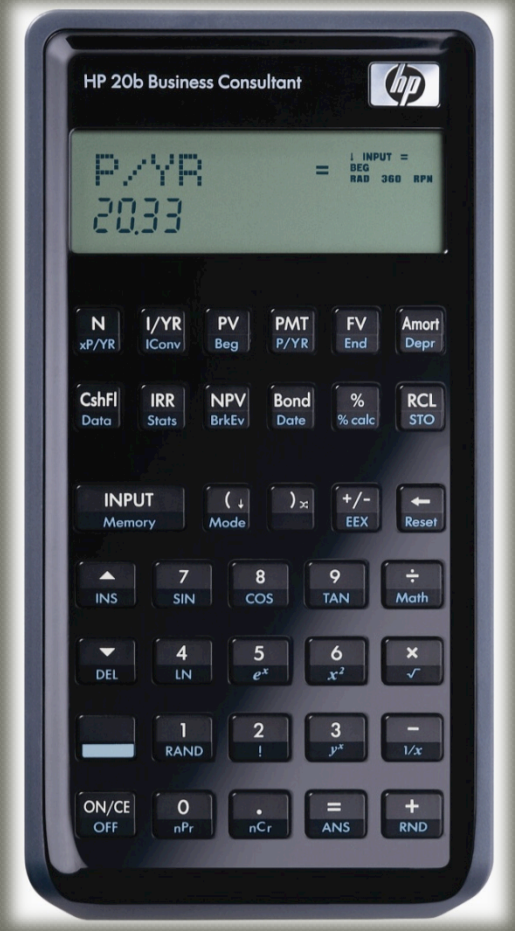
Gateway Lab



RPN Calculator

Michael Yang
Charlie Wu

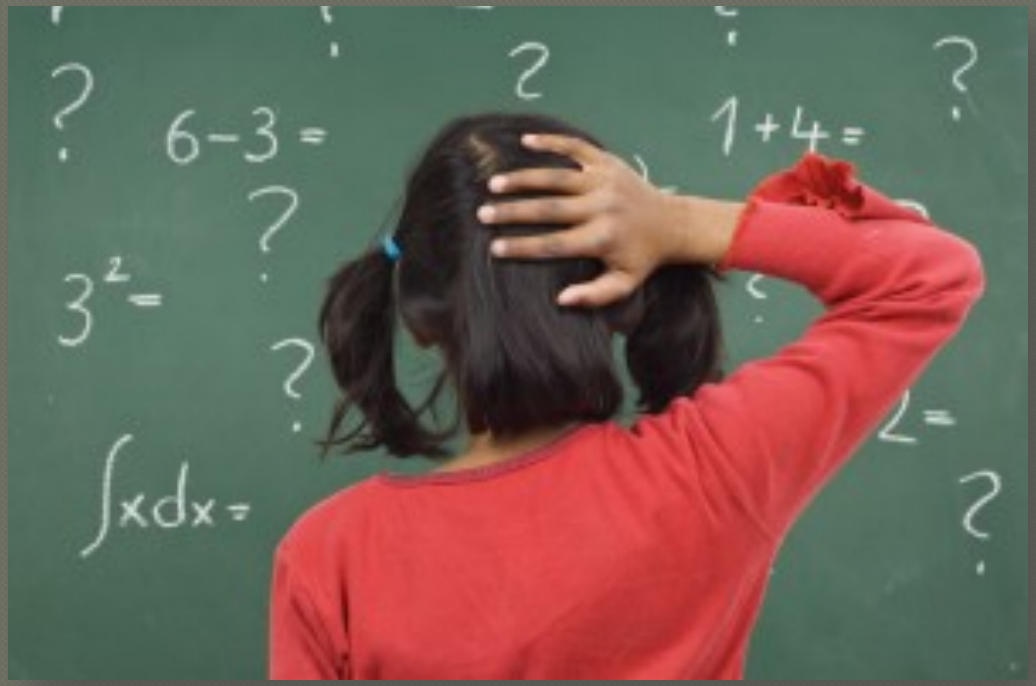
USER GUIDE

Picture of LEGO crawler town courtesy of gadjesteria.com

Interface


HP 20b Business Consultant
P/YR
20.33
= INPUT =
DEG
RAD 360 RPN
N xP/YR
I/YR IConv
PV Beg
PMT P/YR
FV End
Amort Depr
CshFl Data
IRR Stats
NPV BrkEv
Bond Date
% % calc
RCL STO
INPUT Memory
(Mode
) x
+/- EEX
Reset
INS
7 SIN
8 COS
9 TAN
÷ Math
DEL
4 LN
5 e^x
6 x^2
× √
1 RAND
2 !
3 y^x
− 1/x
ON/CE OFF
0 nPr
. nCr
= ANS
+ RND

What is RPN?



Picture courtesy of findmathtutor.com

What is a **stack**?

Picture of 25-stack-IHOP-pancake-tower courtesy of foodbeast.com

Example Calculations

INFIX	RPN
2+4	2 input 4 +
3+6-2	Two methods: 1. 3 input 6 + 2 – 2. 3 input 6 input 2 - +

Example Calculations cont.

- Infix: 3+2*4-8
- RPN: 3 input 2 input 4 * 8 - +

```c
#define MAX_POS 11
#define BASE 10

void displayInt(int number)
{
    int positionLimit = MAX_POS;

    if(number < 0)
    {
        // leave space for '-'
        positionLimit--;
        // get absolute value
        number *= -1;
    }

    int position = 0;

    // continues while digits and spaces are left (or number is initially 0)
    while((number != 0 && position <= positionLimit) || position == 0)
    {
        // get least sig digit
        int digit = number % BASE;

        // print character form of digit

        lcd_put_char7('0' + digit, MAX_POS - position);

        // cut least sig digit
        number /= BASE;
        position++;
    }

    // positionLimit is MAX_POS if positive, MAX_POS - 1 if negative
    if(positionLimit == MAX_POS - 1)
        lcd_put_char7('-', MAX_POS - position);
}
```

Lab 1

```
int keyboard_key_read(int row, int column)
{
  // reset columns except for target column
  int i;
  for(i = 0; i < MAX_COL; i++)
      keyboard_column_high(i);
  keyboard_column_low(column);

  // return if the row is low
  return !keyboard_row_read(row);
}

int keyboard_key()
{
  // initialize to no key
  int coordinate = NO_KEY;

  // check each key in keyboard matrix
  int column, row;
  for(row = 0; row < MAX_ROW; row++)
      for(column = 0; column < MAX_COL; column++)
          if(keyboard_key_read(row, column))
              // return a code containing pressed row and column
              coordinate = 10 * row + column;

  return coordinate;
}
```

Lab 2

Lab 3

```
void keyboard_get_entry(struct entry *result)
{
  // initialize to defaults
  int key = NO_KEY;
  int last_key = NO_KEY;
  int sign = 1;
  int num_buffer = 0;
  int count = 0;

  // run while there is no operation
  while(key == NO_KEY)
  {
    // get a key
    key = keyboard_key();

    // ensure a gap between presses
    while(last_key != NO_KEY)
    {
      last_key = key;
      key = keyboard_key();
    }
```

```
    // handle sign change and continue
    if(key == '~')
    {
      // flip sign
      sign = -1 * sign;

      // handle LCD changes
      if(count == 0)
      {
        if(sign == -1)
          lcd_put_char7('-', LCD_LAST_COLUMN);
        else
          lcd_put_char7(' ', LCD_LAST_COLUMN);
      }
      else
        lcd_print_int_neg(sign == -1, num_buffer);

      // continue
      last_key = key;
      key = NO_KEY;
    }
```

Lab 3 cont.

```
// handle backspaces if there are digits
if(key == '\b' && count > 0)
{
  // cut least significant digit
  num_buffer /= 10;
  count--;

  // handle LCD changes
  if(count == 0)
  {
    if(sign == -1)
    {
      lcd_put_char7(' ', LCD_LAST_COLUMN - 1);
      lcd_put_char7('-', LCD_LAST_COLUMN);
    }
    else
      lcd_put_char7(' ', LCD_LAST_COLUMN);
  }
  else
    lcd_print_int_neg(sign == -1, num_buffer);

  // continue
  last_key = key;
  key = NO_KEY;
}
```

```
  // handle digits
  if(key >= '0' && key <= '9')
  {
    // ensure no overflow
    if(count < MAX_COUNT)
    {
      // append digit
      num_buffer = (num_buffer * 10) + (int)(key - '0');
      count++;
    }

    // handle LCD changes
    lcd_print_int_neg(sign == -1, num_buffer);

    // continue
    last_key = key;
    key = NO_KEY;
  }

  // if none of the above were processed, exit
}

// marker for no number is INT_MAX
if(count == 0)
    num_buffer = INT_MAX;

// loop exits when operation is pressed
// key contains the operation
result->operation = key;
result->number = sign * num_buffer;
```

```
void calculator_rpn(void)
{
  // initialize stack
  int stack[STACK_SIZE];
  int head = 0;

  int last_key = NO_KEY;

  struct entry entry;

  while(1)
  {
    keyboard_get_entry(&entry);

    // handle valid input (non-empty num, op pair)
    if(entry.operation != NO_OP)
    {
      if(entry.number != INT_MAX)
      {…
      }

      // perform non-input operation with check for underflow
      if(entry.operation != INPUT && head != 1)
      {…
      }

      // clear
      if(entry.operation == CLEAR)
      {…
      }

      // continue
      entry.operation = NO_OP;
    }

    // ensure a gap
    while(last_key != NO_KEY)
      last_key = keyboard_key();
  }
}
```

Lab 4

Lab 4 cont.

```
if(entry.number != INT_MAX)
{
  // handle LCD output
  lcd_print_int(entry.number);
  if(entry.operation == INPUT)
    lcd_put_char7('i', 0);
  else
    lcd_put_char7(entry.operation, 0);

  // handle overflow
  if(head == STACK_SIZE)
  {
    //clear stack
    head = 0;
    clear_lcd();

    // print error
    lcd_put_char7('O', 0);
  }

  // push number onto stack
  stack[head++] = entry.number;
}
```

Lab 4 cont.

```
// perform non-input operation with check for underflow
if(entry.operation != INPUT && head != 1)
{
  // pop two entries from stack
  int val1 = stack[--head];
  int val2 = stack[--head];



  int result = 0;

  // perform operation
  switch(entry.operation)
  {
    case '+':
      if (val2/2 + val1/2 <= INT_MAX/2 || -1 * (val2/2 + val1/2) <= INT_MAX/2)
        result = val2 + val1;
      else
      {...
      }
      break;
    case '-':
      if (val2/2 - val1/2 <= INT_MAX/2 || -1 * (val2/2 - val1/2) <= INT_MAX/2)
        result = val2 - val1;
      else
      {...
      }
      break;
    case '*':
      result = val2 * val1;
      break;
    case '/':
      result = val2 / val1;
      break;
    default:
      break;
  }

  // push result onto stack and print
  stack[head++] = result;
  lcd_print_int(result);

  last_key = entry.operation;
}
```

QUESTIONS?

References

- http://www.cs.columbia.edu/~sedwards/classes/2012/gateway-spring/index.html
- http://foodbeast.com/content/2010/08/23/25-stack-ihop-pancakes/
- http://gadgetsteria.com/2010/02/22/lego-junkies-meet-your-master-lego-crawler-town/
- http://findmathtutor.com/