Code Review for Lab 3

Stephen A. Edwards

Columbia University

Fall 2011

```
// Dimitri Dyatlov, Kevin Roark, Nick Duckwiler
#define NUM_OPERATIONS 6
const char operations[] = {'\r','/','*','-','+','='};

oid keyboard_get_entry(struct entry *result)
{
int i, last_key, key_before_last_key, num = 0, count = 0,
    poscheck = 1;

for(;;){
        last_key=keyboard_key();
        if(last_key<'0'+10 && last_key>='0' && (num)<INT_MAX/10 &&
           last_key!=key_before_last_key){ // Checks if input is a
                                           // number.
                lcd_put_char7(last_key,count++); // Prints said number
                                                 // to screen.
                num=num*10+(last_key-'0'); // Num is used to store the
                                           // integer value of our
                                           // input.
        }
```

```
if(last_key=='\b' && last_key!=key_before_last_key &&
   count>0){ // Checks if input is backspace key.
        lcd_put_char7(' ',--count);
        num=num/10; // Drops last digit of integer.
        if (poscheck==-1 && count==1){  // If the negative
                                        // sign is backspaced,
                                        // makes integer
                                        // positive.
                count--;
                poscheck=1;
        }
}
if(last_key=='~' && num==0 &&
   last_key!=key_before_last_key){ // Checks if input is '+/-'
                                   // key.
        if (poscheck==1){
                lcd_put_char7('-',count++);
                poscheck=-1;
        }
        else if (poscheck==-1){
                lcd_put_char7(' ',--count);
                poscheck=1;
        }
}
```

```
        for(i=0;i<NUM_OPERATIONS;++i){
                if(last_key==operations[i]){  // Checks if input is
                                              // one of the operation
                                              // keys. If so, we leave
                                              // the function.
                        lcd_put_char7(last_key,11);
                        result->operation=last_key;
                        num=num*poscheck;
                        result->number=num;
                        return;
                }
        }
key_before_last_key=last_key; // Used to prevent issues of holding a
                              // button for too long, etc.

}
}
```

```
//Ankita Gore, Shikhar Kumar, Christina Huang
void keyboard_get_entry(struct entry *result)
{
result->number = 0; // entry.number
int a = -1;  // nothing is being pressed
int numOfKeysPressed = 0;
int sign = 1; // sign is positive
int position = 1;
for (;;) {
        int b = a;  // b stores previous value of a
        a = keyboard_key();
         /* Report that user entered a digit */
        if ((a=='1' || a=='2' || a=='3' || a=='4' || a=='5' ||
             a=='6' || a=='7' || a=='8' || a=='9' || a=='0') &&
            b==-1) {
                lcd_put_char7(a, position);     // displays key
                                                // pressed
                position++;
                numOfKeysPressed++;             // counts how many
                                                // keys are pressed
                //-----------------------------------------
                int num_to_add = a - '0';      // converts to key
                                               // pressed to int
                // adds to entry.number
                result->number = (result->number)*10 + num_to_add;
        }
```

```
/* Report that user pressed neg/pos button */
if (a=='~' && b==-1) {
        if (sign==-1) { // changes sign displayed when +/- key
                        // is pressed
                lcd_put_char7('-', 0);
        }
        else {
                lcd_put_char7(' ', 0);
        }

        //-----------------------------------------
        sign = sign*-1;          // if neg/pos button pressed,
                                 // then the sign of 'sign'
                                 // switches
}
```

```
            /* entered an operation or INPUT */
            if (a=='/' || a=='*' || a=='+' || a=='-' || a=='\r') {
                    if(numOfKeysPressed == 0) {
                            result->number = INT_MAX; // if no keys
                                                      // pressed, return
                                                      // INT_MAX
                    }
                    if (a=='/' || a=='*' || a=='+' || a=='-') {
                            lcd_put_char7(a, position);
                    }

                    //-----------------------------------------------
                    result->operation = a; // what absolute number should
                                           // be displayed
                    if (sign==-1) {
                            result->number*=-1;  // if sign button is
                                                 // pressed odd number of
                                                 // times, switch
                                                 // entry.number to
                                                 // negative
                    }
                    return;    // if operation/input pressed, return
                               // numbers entered and correct sign
            }
    }
}
```

```
//Feifei Kong, Nicole Lewis, Vanshil Shah
define MAX_DIGITS 9 //can't have a number bigger than 9 digits
#define LAST_SPOT 11

void keyboard_get_entry(struct entry *result)
{
  int c;
  int num=0;  //The immediate number pressed
  int finalvalue=0;     //The entire number that has been entered
  char op;
  int count=0; //spaces away from end of display that number is display
  int value; //value of number still to be printed
  char disp; //actual digit printed
  int digits=0; //number of digits entered
  int neg=0; //keeps track of if it is negative or not

  for(;;)
    {
      c = keyboard_key();
```

```
if(c>='0'&&c<='9') //Checks if the input is a number
  {
    if (!(digits==0&&c=='0')) { //does not do anything if the
                                //first number pressed is 0
      if (digits<MAX_DIGITS) { //makes sure number doesn't
                               //have more than 9 digits
        num=c-'0';
        finalvalue = finalvalue*10 + num;
        value=abs(finalvalue);
        count=0; //represents the spaces away from the right end
                 //of the display
        while(value>0) {
          disp=(value%10)+'0'; //character to be displayed
          lcd_put_char7(disp,LAST_SPOT-count); //puts it in the
                                               //spot count
                                               //number away
                                               //from the end

          value=value/10;
          count++;
        }
        while(c!=-1) { //wait until key is released
          c=keyboard_key();
        }
        digits++;
      }
    }
```

```
} else if (c!=-1) { //if something other than a number is
                    //pushed
if (c=='~') { //if the negative sign is pushed
  finalvalue=finalvalue*-1; //makes final value the opposite
                            //sign
  if (neg==0) { //if it was not negative before, makes
                //negative sign appear and neg true
    lcd_put_char7('-',0);
    neg=1;
  } else { //if neg was true before, makes negative sign go
           //away and neg false
    lcd_put_char7(' ',0);
    neg=0;
  }
  while(c!=-1) {
    c=keyboard_key();
  }
```

```
} else if(c=='\b') { //if backspace key is pressed after
                     //number have begun to be entered
  if (digits>0) { //checks that numbers have begun to be entere
    finalvalue=finalvalue/10; //gets rid of last digit
    //reprint everything:
    value=abs(finalvalue);
    count=0;
    while(value>0) {
      disp=(value%10)+'0'; //character to be displayed
      lcd_put_char7(disp,LAST_SPOT-count); //puts it in the
                                           //spot count number
                                           //away from the end
      value=value/10;
      count++;
    }
    lcd_put_char7(' ',LAST_SPOT-count);//cover up the spot at
                                       //the very front
    digits--;
    while(c!=-1) {
      c=keyboard_key();
    }
  }
```

```
        } else {
          if (digits=0) { //no number was entered
            finalvalue=INT_MAX;
          }
          op=c; //Once a non-number is pressed records
          lcd_put_char7(op,LAST_SPOT+1);
          break;
        }
      }

    }
  result->number = finalvalue;      //Reports the final value of the nu
  result->operation = op;  //Reports the operation that is pressed
}
```

```
//Andrew Pope, Will VanArsdall, Abhinav Mishra

char *intToStr(int number)
{
  int i, j;
  int temp = number;

  //Counts digits in number
  for(i = 0;;i++)
  {
    if(temp == 0)
      break;
    temp /= NUM_BASE;
  }

  //Stores string after function return
  static char str[COLUMNS + 1];

  //Clears string between function calls
  for(j = 0; j < COLUMNS; j++)
  {
    str[j] = ' ';
  }
  str[COLUMNS] ='\0';
```

```
  //Iterates through number converting digits to characters
  for(j = COLUMNS - 4; j > COLUMNS - i - 4; j--)
  {
    int digit = number % NUM_BASE;
    digit += '0';
    str[j] = (char) digit;
    number /= NUM_BASE;
  }

  return str;
}

int pow(int num, int p)
{
  int ret = 1;
  int i;

  if(p != 0)
    for(i = 0; i < p; i++)
      ret *= num;
  return ret;
}
```

```
void keyboard_key_entry(struct entry *result)
{
  char num[COLUMNS];
  char prevKey;
  int pos = DIGIT_COLUMNS;
  int i, prevPos;

  //Fills printable columns to remove junk data
  for(i = 0; i < COLUMNS; i++)
  {
    num[i] = ' ';
  }
```

```
for(;;)
{
  int input = keyboard_key();
  char key = (char) input;

  if((key == '+' || key == '*' || key == '/' || key == '-' ||
      key == '=' || key == '\r') && key != prevKey)
  {
    result->operation = key;
    int val = 0;
    int numDigits = DIGIT_COLUMNS - pos;
    if(pos == DIGIT_COLUMNS)
      val = INT_MAX;
    else
      for(i = 1; i <= numDigits; i++) //Iterates through number
                                      //string, converting digits to
                                      //their numerical values
      {
        if(num[pos + i] != ' ')
        {
          int digit = num[pos + i] - '0';
          val += digit * pow(NUM_BASE, numDigits - i);
        }
      }
    result->number = val;
    break;
  }
```

```
else if((key == '0' || key == '1' || key == '2' || key == '3' ||
         key == '4' || key == '5' || key == '6' || key == '7' ||
         key == '8' || key == '9') && key != prevKey)
{
   if(pos == DIGIT_COLUMNS) //Places digit in the first column on
                            //first run
   {
     num[pos] = key;
     prevPos = pos;
     pos--;
   }
   else
   {
     for(i = pos + 1; i <= DIGIT_COLUMNS; i++) //Shifts numbers on
                                               //the screen back
                                               //one space
     {
       char temp = num[i];
       num[i - 1] = temp;
     }
     num[DIGIT_COLUMNS] = key;
     prevPos = pos;
     pos--;
   }
}
```

```
    //Printing
    if(pos != DIGIT_COLUMNS && pos != prevPos && input != -1)
      lcd_print7(num);

    if(input == -1)
      prevKey = ' ';
    else
      prevKey = (char) input;
  }
}
```

```
//By Kaiven Zhou, Yiming Ge, Anna Teng

#define MAX_NUM_DIGITS 11

void keyboard_get_entry(struct entry *result)
{
        int inputNumber=INT_MAX;

        int temp=-1,inputOperation=-1;
        int i=0;
        for(; i<MAX_NUM_DIGITS; i++)
        {
                while(temp==-1) //keeps going until a key is
                                //inputed. Pressed key stored in temp
                {
                        temp = keyboard_key();
                }
                if(temp=='0'||temp=='1'||temp=='2'||temp=='3'||
                   temp=='4'||temp=='5'||temp=='6'||temp=='7'||
                   temp=='8'||temp=='9')
                {
                        if(inputNumber==INT_MAX)
                                inputNumber=0;
                        int integerOfTemp = temp-'0'; //convert from
                                                      //char to int by
                                                      //subtracting
                                                      //ascii value of
                                                      //zero
```

```
                inputNumber*=10;
                inputNumber+=integerOfTemp;

                lcd_put_char7(temp,i);
        }
        else
        {
                inputOperation = temp;
                break;
        }

        while(temp!=-1) //Calculator going too fast. Without
                        //this, I'll still be pressing the key
                        //when the for loop iterates
        {
                temp = keyboard_key();
        }

}
```

```
        while(1)
        {
                if(inputOperation=='+'||inputOperation=='-'||
                   inputOperation=='*'||inputOperation=='/'||
                   inputOperation=='\r')
                {
                  //display operation at the last placed character
                        lcd_put_char7(inputOperation,MAX_NUM_DIGITS);
                        break;
                }
                else
                {
                        inputOperation = keyboard_key();
                }
        }

  result->number = inputNumber;
  result->operation = inputOperation;
}
```

```
// Stephen Edwards

#  define LCD_7_COLUMNS 15

// Display a right-justified integer with an optional
// leading negative sign
void lcd_print_int_neg(int negative, unsigned int n)
{
  int column = LCD_7_COLUMNS - 4;           // Start at right
  do {                                      // Always at least one
    lcd_put_char7(n % 10 + '0', column--); // Display a digit
    n /= 10;                                // Go to next digit
  } while (n);
  if (negative)       lcd_put_char7('-', column--); // Neg sign?
  while (column >= 0) lcd_put_char7(' ', column--); // Clear to left
}

// Display a right-justified signed integer
void lcd_print_int(int n) {
  if (n < 0) lcd_print_int_neg(1, -n);
  else       lcd_print_int_neg(0, n);
}

// Return a character indicating which key was pressed or 0
// See keyboard.c for which character is returned by each key
extern int keyboard_key(void);
// (Return value changed)
```

```
#define DECIMAL_INT_MAX 100000000
void keyboard_get_entry(struct entry *result)
{
  int key;
  unsigned int entry = INT_MAX;                 // Number being entered
  int negative = 0;                             // Positive by default
  for (;;) {
    while (keyboard_key()) ;                    // Wait until no key pressed
    while (!(key = keyboard_key())) ;           // Wait for a keypress
    if ( key >= '0' && key <= '9') {
      if (entry == INT_MAX) entry = 0;          // First digit?
      if (entry < DECIMAL_INT_MAX)              // avoid overflow
        entry = entry * 10 + (key - '0');       // Add digit to right
    } else if (key == '~') {                    // +/- key
      if (entry == INT_MAX) entry = 0;          // First press gives "-0"
      negative = !negative;                     // Toggle sign
    } else if (key == '\b' && entry != INT_MAX) // Backspace
      entry /= 10;                              // Discard rightmost digit
    else if (key == '\r' || key == '+' || key == '-' ||
             key == '*'  || key == '/') {
      result->number = negative ? -entry : entry;
      result->operation = key;
      return;
    }
    lcd_print_int_neg(negative, entry);
  }
}
```