```
/*
p-1.c
        p-1.c
        Kimball Martin
        CMSC 443
        5/1/97

        C implementation of Pollard's p - 1 method using Henrik Johansson's bignum
        code.  Code was written for numbers with up to 256 bits but this may easily
        be modified by redefining MAX_BIT (though you probably wouldn't want to run
        this on a number so large).
        name code p-1.c
        To compile:
                cc p-1.c bignum.c -o p-1
        Usage:
                p-1 <numfile>
        where numfile is the file containing just the number.

        Modified by Brooke Stephens....spring 1997
        Note -- this code may still have bugs in it.
*/



#define MAX_BIT 256             /* Maximum number of bits in integer to factor */
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "bignum.h"

/* Function to do one iteration of the p - 1 method.
   Takes in n, the number to factor,
            a, the number to exponentiate, and
            B, the upper bound for the factor base.
*/
void pminus1(bignum, bignum, bignum);

void main(int argc, char *argv[]) {

  FILE *f;
  char c;
  char *s;
  bignum n, a, B;

  if (argc!=2) {
    printf("Usage: %s <filename>\n",argv[0]);
    exit(1);
  }

  if((f = fopen(argv[1], "r")) == NULL) {
    printf("Error opening file!\n");
    exit(2);
  }

  big_init_pkg();                                       /* Bignum initialization */
  big_create(&n);
```

```
  big_create(&B);
  big_create(&a);

  s = (char *) malloc(sizeof(char) * (MAX_BIT + 1));    /*Read in number to factor
*/
  fgets(s, MAX_BIT, f);
  big_set_string(s, 10, &n);
  printf("The number to factor is:\n%s\n",s);

  while (tolower(c) != 'n') {                           /*Interactively let the */
    fflush(stdin);                                      /*user choose the bound */
    printf("Enter the bound B: ");
    fgets(s, MAX_BIT, stdin);
    big_set_string(s, 10, &B);
    fflush(stdin);
    printf("Enter an initial value between 2 and n-2: ");
     fgets(s, MAX_BIT, stdin);                  /* and the number to */
     /* big_set_long(2,&a); */
    big_set_string(s, 10, &a);                          /* exponentiate */
    pminus1(n, a, B);
    fflush(stdin);
    printf("Continue [y/n]?");
    scanf("%c", &c);
  }

  big_destroy(&n);                                      /*Clean up bignum */
  big_destroy(&B);
  big_destroy(&a);
  big_release_pkg();

}

void pminus1 (bignum n, bignum a, bignum B) {
  bignum d, k, i, b,one, two;

  big_create(&d);               /*Initialize bignum vars */
  big_create(&k);
  big_create(&i);
  big_create(&b);
  big_create(&one);
  big_create(&two);
  big_set_long(1, &k);                                  /* k = 1 */
  big_set_long(1, &one);
  big_set_long(2, &two);
  big_set_long(2, &i);
  big_set_long(&a, &b);
  big_set_long(1, &d);
  printf("a %s \n ", big_string(&a,10));
  printf("entering p-1\n");                     /* k = B! */
  /* for (big_set_long(2, &i); big_leqp(&i, &B);big_add(&i,&one,&i))*/
  while (big_leqp(&d,&one))
   { big_exptmod(&a,&i,&n,&a);
    /* printf("%s\n  ",big_string(&a,10)); */



    /*printf(big_string(&a,10)); */                                   /* i
```

```
 = a^k mod n*/
   big_sub(&a, &one, &b);
   big_gcd(&b, &n, &d);                          /* d = gcd(a^k - 1, n) */
   if (big_lessp(&one, &d) && big_lessp(&d, &n)) {       /* if d is nontrivial */
     printf("n factors into %s ", big_string(&d,10));
     big_round(&n,&d,&b,&i);                             /* a = n/d with remainder
 i*/
     printf("and %s\n", big_string(&b,10));              /*    (i will be 0) */

   };
    big_add(&i,&one,&i);
    };
   /*  else
     { printf("The p-1 method failed.  Try choosing a larger bound.\n");
     printf(big_string(&d,10)); */

   big_destroy(&d);                                    /* cleanup bignum */
   big_destroy(&k);
   big_destroy(&i);
   big_destroy(&one);
 }
```