Lecture 16: Programming in PARI, II

William Stein

Math 124 Harvard University **Fall 2001**

1 Beyond One Liners

In today's relaxing but decidely non-mathematical lecture, you will learn a few new PARI programming commands. Feel free to try out variations of the examples below (especially because there is no homework due this coming Wednesday). Also, given that you know PARI fairly well by now, ask me questions during today's lecture!

1.1 Reading Files

The `\r` command allows you to read in a file.

Example 1.1. Create a file `pm.gp` that contains the following lines

```
{powermod(a, p, n) =
    return (lift(Mod(a,p)^n));}
```

Now use `\r` to load this little program into PARI:

```
> ?powermod
  ***   powermod: unknown identifier.
> \rpm               \\ \rpm.gp would do the same thing
? ?powermod
powermod(a, p, n) = return(lift(Mod(a,p)^n));
? powermod(2,101,7)
%1 = 27
```

If we change `pm.gp`, just type `\r` to reload it (omitting the file name reloads the last file loaded). For example, suppose we change `return (lift(Mod(a,p)^n))` in `pm.gp` to `return (lift(Mod(a,p)^n)-p)`. Then

```
? \r
? powermod(2,101,7)
%2 = -74
```

1.2 Arguments

PARI functions can have several arguments. For example,

```
{add(a, b, c)=
   return (a + b + c);}
? add(1,2,3)
%3 = 6
```

If you leave off arguments, they are set equal to 0.

```
? add(1,2)
%4 = 3
```

If you want the left-off arguments to default to something else, include that information in the declaration of the function:

```
{add(a, b=-1, c=2)=
   return (a + b + c);}
? add(1,2)
%6 = 5
? add(1)
%7 = 2
? add(1,2,3)
%8 = 6
```

1.3 Local Variables Done Right

Amidst the haste of a previous lecture, I mentioned that an unused argument can be used as a poor man's local variable. The following example illustrates the right way to declare local variables in PARI.

Example 1.2. The function `verybad` below sums the integers $1, 2, \ldots n$ whilst wreaking havoc on the variable `i`.

```
{verybad(n)=
   i=0;
   for(j=1,n, i=i+j);
   return(i);}
? verybad(3)
%9 = 6
? i=4;
? verybad(3);
? i
%13 = 6                    \\ ouch!! what have you done to my eye!
```

The function `poormans` is better, but it uses a cheap hack to simulate a local variable.

```
{poormans(n,  i=0)=
   for(j=1,n, i=i+j);
   return(i);}
? i=4;
? poormans(3)
%16 = 6
? i
%17 = 4               \\ good
```

The following function is the best, because i is local and it's clearly declared as such.

```
{best(n)=
   local(i);
   i=0;  for(j=1,n, i=i+j);
   return(i);}
? i=4;
? best(3)
%18 = 6
? i
%19 = 4
```

1.4 Making Your Program Listen

The `input` command reads a PARI expression from the keyboard. The expression is evaluated and the result returned to your program. This behavior is at first disconcerting if, like me, you naively expect `input` to return a string. Here are some examples to illustrate the `input` command:

```
? ?input
input(): read an expression from the input file or standard input.
? s = input();
1+1
? s                     \\ s is not the string "1+1", as you might expect
%24 = 2
? s=input()
hi there
%25 = hithere
? type(s)               \\ PARI views s as a polynomial in the variable hithere
%26 = "t_POL"
? s=input()
"hi there"
%27 = "hi there"
? type(s)               \\ now it's a string
%28 = "t_STR"
```

1.5 Writing to Files

Use the `write` command:

```
? ?write
write(filename,a): write the string expression a to filename.
? write("testfile", "Hello Kitty!")
```

The `write` command above appended the line "Hello Kitty!" to the last line of `testfile`. This is useful if, e.g., you want to save key bits of work during a session or in a function. There is also a **logging facility** in PARI, which records most of what you type and PARI outputs to the file `pari.log`.

```
? \l
   log = 1 (on)
? 2+2
%29 = 4
? \l
   log = 0 (off)
   [logfile was "pari.log"]
```

2 Coming Attractions

The rest of this course is about continued fractions, quadratic forms, and elliptic curves. The following illustrates some relevant PARI commands which will help us to explore these mathematical objects.

```
? ?contfrac
contfrac(x,{b},{lmax}): continued fraction expansion of x ...
? contfrac(7/9)
%30 = [0, 1, 3, 2]
? contfrac(sqrt(2))
%31 = [1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...]
? ?qfbclassno
qfbclassno(x,{flag=0}): class number of discriminant x using Shanks's
method by default. If (optional) flag is set to 1, use Euler products.
? qfbclassno(-15,1)  \\ ALWAYS use flag=1, since ``the authors were too
%32 = 2              \\ lazy to implement Shanks' method completely...''
? E=ellinit([0,1,1,-2,0]);
? P=[0,0];
? elladd(E,P,P)
%36 = [3, 5]
? elladd(E,P,[3,5])
%37 = [-11/9, 28/27]
? a=-11/9;b=28/27;           \\ this is an ``amazing'' point on the curve.
? b^2+b == a^3+a^2-2*a
%38 = 1
```