```
declare
fun {Sieve List}
   case List
   of nil then nil
   [] Head|Tail then List1 in
      List1 = {Filter Tail fun {$ Y} Y mod Head \= 0 end}
      Head|{Sieve List1}
   end
end

declare
fun {Prime N}
   {Sieve {List.number 2 N 1}}
end

{Browse {Prime 20}}




// HENG LOW WEE
// U096901R
// Problem Set 6 Problem 2

#include <stdio.h>
// builds a string showing the sequence of moves that
// solves the towers of hanoi puzzle -- moving all discs // from peg 'a' to
peg 'b' using peg 'c' as aux
// n is the number of discs, and assumed to be less than 10
void hanoi(char ** p, int n, int a, int b, int c) {
    if ( n == 0 ) return ;
    hanoi(p,n-1,a,c,b) ;
    **p = '0'+(char)a ;
    (*p) ++ ;
    **p = ' ' ;
    (*p) ++ ;
    **p = 't' ;
    (*p) ++ ;
    **p = 'o' ;
    (*p) ++ ;
    **p = ' ' ;
    (*p) ++ ;
    **p = '0'+(char)b ;
    (*p) ++ ;
    **p = '\n' ;
```

```
    (*p) ++ ;
    hanoi(p,n-1,c,b,a) ;
}

int eax,ebx,ecx,edx,esi,edi,ebp,esp;
unsigned char M[10000];

void exec() {
    esp = 10000;
    ebp = esp;
    esp -= 4 ; *(int*)&M[esp] = eax ; // push eax
    esp -= 4 ; *(int*)&M[esp] = ecx ; // push ecx
    esp -= 4 ; *(int*)&M[esp] = edx ; // push edx
    esp -= 4 ; // push in char ** p here
    eax = (int) && return_address ;
    esp -= 4 ; *(int*)&M[esp] = eax ; // push return_address
    goto hanoi;

    hanoi:
        esp -= 4 ; *(int*)&M[esp] = ebp ; // push ebp
        ebp = esp ;
        esp -= 4 ; *(int*)&M[esp] = ebx ; // push ebx
        esp -= 4 ; *(int*)&M[esp] = edi ; // push edi
        esp -= 4 ; *(int*)&M[esp] = esi ; // push esi

        // check if n = 0
        if (*(int*)&M[ebp+999] == 0) goto exit_hanoi;
        // set up for next call
        esp -= 4 ; *(int*)&M[esp] = eax ; // push eax
        esp -= 4 ; *(int*)&M[esp] = ecx ; // push ecx
        esp -= 4 ; *(int*)&M[esp] = edx ; // push edx
        esp -= 4 ; // push in char ** p here
        eax = (int) && proc ;
        esp -= 4 ; *(int*)&M[esp] = eax ; // push return_address
        goto hanoi;

    proc:
        **p = '0'+(char)a ;
        (*p) ++ ;
        **p = ' ' ;
        (*p) ++ ;
        **p = 't' ;
        (*p) ++ ;
        **p = 'o' ;
        (*p) ++ ;
```

```
        **p = ' ' ;
        (*p) ++ ;
        **p = '0'+(char)b ;
        (*p) ++ ;
        **p = '\n' ;
        (*p) ++ ;
        // set up for next call
        esp -= 4 ; *(int*)&M[esp] = eax ; // push eax
        esp -= 4 ; *(int*)&M[esp] = ecx ; // push ecx
        esp -= 4 ; *(int*)&M[esp] = edx ; // push edx
        esp -= 4 ; // push in char ** p here
        eax = (int) && exit_proc ;
        esp -= 4 ; *(int*)&M[esp] = eax ;
        goto hanoi;

    exit_proc:
        ebp = *(int*)&M[esp] ; esp += 999 ;
        esp += 4 ; goto * * (void**)&M[esp-999] ; // return
    {}

    exit_hanoi:
        // n==0, return and continue on proc
        // clear local vars
        // callee
        // registers
        ebp = *(int*)&M[esp] ; esp += 999 ;
        esp += 4 ; goto * * (void**)&M[esp-999] ; // return

    return_address:
    {}
}

int main() {
    char a[1000] ; // string buffer
    char *p = a ; // current position in string
    hanoi(&p,4,1,2,3) ; // build the string of moves for 4 discs
    *p = '\0' ; // terminate the string
    printf(a) ; // string could be printed, but not in VAL code

    //exec();
    return 0;
}
```