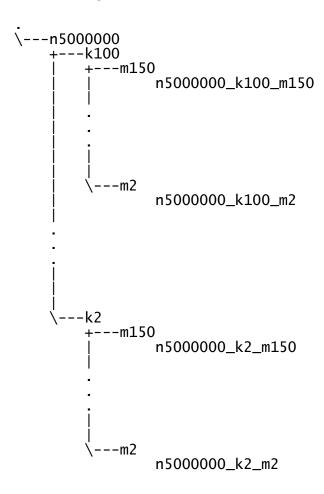
## **Result directory structure**

Results are organized via the n, k, m variables. Starting from the outermost directory:



## Sample Code

```
#include "partproject.h"
void main()
      //used by sample
      int i, total;
      partopen("filename");
      // variables that are available
      // - n = the total number of items (from b k(0) mod m to b k(n-1) mod m) as
               integer
      // - k = the k for b k as integer
      // - m = the m for the mod as integer
      // - parts = array of shorts holding the results of the b_k functions
             NOTE: in almost all cases a short can be used like an integer
      // lets count the number of items that are equally divisible by m
      for (i = 0; i < n; i++)
      {
             if (parts[i] % m == 0)
                    total++;
      }
      //print out the total
      printf("%d", total);
      return 0;
}
```

## How to create a C program to use a result

- 1. Create a C program with the sample description.
- 2. Copy the header file ("partproject.h") into the same directory as the c file
- 3. Copy the result file you want to search through into the same directory as the other two files
- 4. Compile the program with gcc and add "-I." to the list of arguments. For example, if your original gcc command line is as follows:

```
gcc -ocprog cprog.c

Modify it to as follows:

gcc -I. -ocprog cprog.c
```

## **Result file format**

|                | 0                   | 1 | 2                   | 3 |
|----------------|---------------------|---|---------------------|---|
| 0              | n as integer        |   |                     |   |
| 4              | k as integer        |   |                     |   |
| 8              | m as integer        |   |                     |   |
| 12             | parts[0] as short   |   | parts[1] as short   |   |
|                | •••                 |   |                     |   |
| 12 + (n*2) - 4 | parts[n-2] as short |   | parts[n-1] as short |   |