Table 1: Function / Block Level - PART I

F_{c1} - data dependency

```
/* uses global character buffer generated from message_generate function of sockfilt file*/
void win32_perror(const char *msg)
```

F_{c2} - call order

```
/* png_destroy_png_struct calls png_free
which  might call png_error and may certainly call
       * png_get_mem_ptr, so fake a temporary png_struct to support this.
       */
void  png_destroy_png_struct(png_structrp png_ptr)
```

F_{c3} - algorithm outline / working summary

```
/* Allocate memory.  For reasonable files, size should never exceed
 * 64K.  However, zlib may allocate more than 64K if you don't tell
 * it not to.  See zconf.h and png.h for more information.  zlib does
 * need to allocate exactly 64K, so whatever you call here must
 * have the ability to do that.
 */
PNG_FUNCTION(png_voidp,PNGAPI
png_calloc,(png_const_structrp png_ptr, png_alloc_size_t size),PNG_ALLOCATED)
{
```

```
/* Free a pointer allocated by png_malloc().  If ptr is NULL, return
 * without taking any action.
 */
void PNGAPI
png_free(png_const_structrp png_ptr, png_voidp ptr)
{
```

F_{c4} - mapping to AD

```
//  used extensively in resetting standard x-y plots, semi-log plots
void
plFree2dGrid( PLFLT **f, PLINT nx, PLINT PL_UNUSED( ny ) )
{
```

F_{c5} - description of the dataset or global data stores being used

```
  # works on a two dimensional data matrix (each of size 8) generated from light rider bot module
    def flood_fill(self, position, visited):
```

```
/* pngmem.c - stub functions for memory allocation of libpng -- raster-graphics file-format*/
#include "pngpriv.h"
```

F_{c6} - links to project management details

```
#if defined(PNG_TEXT_SUPPORTED) || defined(PNG_sPLT_SUPPORTED) ||\
   defined(PNG_STORE_UNKNOWN_CHUNKS_SUPPORTED)
/* Introduced in libpng 1.6.0, commit 0e13545. This is really here only to work round a spurious
    warning in GCC 4.6 and 4.7
 * that arises because of the checks in png_realloc_array that are repeated in
 * png_malloc_array, issue #289.
 */

static png_voidp
png_malloc_array_checked(png_const_structrp png_ptr, int nelements,
    size_t element_size)
{
```

Table 2: Function / Block Level – PART II

F_{c7} - descriptions of parameters / return type

```
//! Allocate a block of memory for use as a matrix of type
//! PLFLT_MATRIX (organized as an Iliffe column vector of pointers to
//! row vectors).  As a result the matrix can be accessed using C/C++
//! syntax like *f[i][j]. The memory associated with this matrix must
//! be freed by calling plFree2dGrid once it is no longer required.
//! Example usage:
//!
//!   PLFLT **z;
//!
//!   plAlloc2dGrid(&z, XPTS, YPTS);
//!
//! @param f Location of the storage (address of a **).
//! @param nx Size of the grid in x.
//! @param ny Size of the grid in y.
//!
//--------------------------------------------------------------------------

void
plAlloc2dGrid( PLFLT ***f, PLINT nx, PLINT ny )
{
```

F_{c8} - possible exceptions

```
   /* Check for overflow on the elements count (so the caller does not have to
    * check.) png_malloc_array has been worked with the size calculations to avoid
          * overflow.
    */
PNG_FUNCTION(png_voidp,
png_realloc_array,(png_const_structrp png_ptr, png_const_voidp old_array,
    int old_elements, int add_elements, size_t element_size),PNG_ALLOCATED)
{
   /* These are internal errors: */
```

F_{c9} - description of external libraries used

```
  /* uses png_calloc defined in pngriv.h*/
PNG_FUNCTION(png_voidp,PNGAPI
png_calloc,(png_const_structrp png_ptr, png_alloc_size_t size),PNG_ALLOCATED)
{
```

F_{c10} - markers - namespace, macros, class, function

```
 }
#endif /* TEXT || sPLT || STORE_UNKNOWN_CHUNKS */
```