```c
// Author: i15fujimura1s@tokuyama.ac.jp
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "pnm.h"

static void skipComments(FILE *fp) {
  int c;
  while((c = fgetc(fp)) == '#') {
    while(c != '\n') {
      c = fgetc(fp);
    }
  }
  ungetc(c, fp);
}

static void skip(FILE *fp, const char *delim) {
  int c;
  while(strchr(delim, c = fgetc(fp)));
  ungetc(c, fp);
}

static void skipDelimiters(FILE *fp) {
  int c;
  while(strchr(" \n", c = fgetc(fp)))
    if(c == '\n') skipComments(fp);
  ungetc(c, fp);
}

static int readBytesAsInt(FILE *fp, int unit) {
  int value = 0;
  for(int i = 0; i < unit; i++) {
    value <<= 8;
    value |= fgetc(fp);
  }
  return value;
}

static int readInt(FILE *fp) {
  int value = 0, c;
  while(strchr("0123456789", c = fgetc(fp)))
    value = value * 10 + (c - '0');
  ungetc(c, fp);
  return value;
}

const char *readHeader(FILE *fp, PNM *pnm) {
  skipComments(fp);
  if(fgetc(fp) != 'P') return "bad signature";
  int desc = readInt(fp);
  if(desc < 1 || desc > 6) return "invalid magic number";
  pnm->descriptor = desc;
  skip(fp, " \n");
  pnm->width = readInt(fp);
  skip(fp, " \n");
  pnm->height = readInt(fp);
  if(desc == 1 || desc == 4) {
    pnm->max = 1;
  } else {
    skip(fp, " \n");
    pnm->max = readInt(fp);
  }
  skip(fp, " ");
  if(fgetc(fp) != '\n') return "bad header line";

  pnm->count = pnm->width * pnm->height;
  if(desc == 3 || desc == 6) pnm->count *= 3;

  return NULL;
}

const char *readPNM(FILE *fp, PNM *pnm) {
  if(!pnm) return "pnm is NULL";

  const char *error;
  if((error = readHeader(fp, pnm)) != NULL) return error;
  unsigned short *data = (unsigned short *) malloc(sizeof(unsigned short) * pnm->count);
  int count = 0;
  switch(pnm->descriptor) {
    case 1:
    case 2:
    case 3:
    {
      while(count < pnm->count && !feof(fp)) {
        skipDelimiters(fp);
        data[count++] = readInt(fp);
      }
      break;
    }
    case 4:
    {
      while(count < pnm->count && !feof(fp)) {
        int b = fgetc(fp);
        for(int i = 0; i < 8; i++) data[count++] = (b & (1 << i)) != 0;
      }
      break;
    }
    case 5:
    case 6:
    {
      int unit = pnm->max < 256 ? 1 : 2;
      int length = unit * pnm->count;
      while(count < pnm->count && !feof(fp)) {
        data[count++] = readBytesAsInt(fp, unit);
      }
      break;
    }
  }
  pnm->count = count;
  pnm->data = data;

  return NULL;
}

void freePNM(PNM *pnm) {
  if(pnm->data) {
    free(pnm->data);
    pnm->data = NULL;
  }
}
```