1. Frame file plotter. Open a frame file, read it's contents, and produce a PostScript plot.

```
#define SAMPLERATE 16000
#include <stdio.h>
#include <math.h>
  char *tag_strings[] ← {"␣",      /* Null Case */
  "aI","aU","eI","oI","oU","3R",      /* Dipthongs */
  "tS","dZ","D","N","S","T","Z",      /* Complex Consonants */
  "&","A","e","i","I","O","u","U","V",      /* Vowels */
  "b","d","f","g","h","j","k","l","m",      /* Consonants */
  "n","p","r","s","t","v","w","z",      /* Consonants */
  (char *) Λ      /* Null ptr signals end */
  };
  main(int argc, char *argv[])
  {
    FILE *inf;
    int nframes, framesize;
    char *s, the_line[128];
    float one_value;
    int targetval;
    int i, j;
    double litvalue, samples, samples_per_point;
    int *targetvals, tbeg, tend, lab;
    argc−−;
    argv++;
    if (¬argc) {
      fprintf(stderr, "No␣arguments!␣␣I␣need␣a␣frame␣file.\n");
      exit(1);
    }
    if ((inf ← fopen(*argv, "r")) ≡ Λ) {
      fprintf(stderr, "couldn't␣open␣file␣\"%s\"\n", *argv);
      exit(1);
    }
    fgets(the_line, sizeof (the_line), inf);
    if (strncmp(the_line, "framesize␣=", 11) ≠ 0) {
      fprintf(stderr, "Frame␣file␣has␣bad␣format:␣missing␣framesize\n");
      exit(1);
    }
    s ← the_line + 11;
    framesize ← atoi(s);
    fgets(the_line, sizeof (the_line), inf);
    if (strncmp(the_line, "nframes␣=", 9) ≠ 0) {
      fprintf(stderr, "Frame␣file␣has␣bad␣format:␣missing␣nframes\n");
      exit(1);
    }
    s ← the_line + 9;
    nframes ← atoi(s);
    targetvals ← (int *) malloc(sizeof(int) * nframes);      /* Print out the PostScript header */
    printf("%%!\n");
    printf("/maxval␣1.0␣def\n");
    printf("7.75␣72␣mul␣0.75␣72␣mul␣translate␣90␣rotate\n");
    printf("/frames␣%d␣def\n", nframes);
```

```
printf("/bh␣7.25␣72␣mul␣%d␣div␣def\n", framesize);
printf("/bw␣10␣72␣mul␣frames␣div␣def\n");
printf("/x␣0␣def␣/y␣0␣def\n");
printf("/d␣{␣maxval␣mul␣neg␣1␣add␣setgray\n");
printf("␣x␣y␣moveto␣0␣bh␣rlineto␣bw␣0␣rlineto␣0␣bh␣neg␣rlineto\n");
printf("␣closepath␣fill␣/y␣y␣bh␣add␣def␣}␣bind␣def\n");
printf("/r␣{␣/y␣0␣def␣/x␣x␣bw␣add␣def␣}␣def\n");    /* Read the rest of the frame file, spitting
    out the data as we go. Save the target values for annotating the output later. */
for (i ← 0; i < nframes; i++) {
  fscanf(inf, "%d\n", &(targetvals[i]));
  for (j ← 0; j < framesize; j++) {
    fscanf(inf, "%g\n", &one_value);
    printf("%.3g␣d\n", one_value);
  }
  printf("r\n");
}    /* I want to draw lines on the chart at intervals which are some factor of 10, which will indicate
      either seconds or milliseconds. First, determine the number of samples per PostScript point on
      the chart. */
samples_per_point ← (double)(nframes * framesize * 2)/(double)(10 * 72);
  /* Now determine the approximate number of seconds represented by 1/2 an inch. */
samples ← 36.0 * samples_per_point/(double) SAMPLERATE;
  /* Find the power of ten represented by this value. If less than a second, we will display the number
    of milliseconds. If more than a second, we will display seconds. */
i ← 0;
litvalue ← 1.0;
while (samples < 1.0) {
  i++;
  samples *= 10.0;
  litvalue /= 10.0;
}
while (samples > 10.0) {
  i--;
  samples /= 10.0;
  litvalue *= 10.0;
}    /* Round the value up to the next closest integer factor. */
samples ← ceil(samples);
litvalue *= samples;    /* Restore the power value. */
while (i > 0) {
  samples /= 10.0;
  i--;
}
while (i < 0) {
  samples *= 10.0;
  i++;
}    /* Calculate the number of samples represented by the selected time slice. */
samples *= SAMPLERATE;    /* Output the number of points represented by each time slice. */
printf("/bw␣%.3lf␣def␣/x␣0␣def␣0␣setgray␣.3␣setlinewidth\n", samples/samples_per_point);
printf("/Helvetica␣findfont␣8␣scalefont␣setfont\n");
printf("/cs{moveto␣dup␣stringwidth␣pop␣-2␣div␣0␣rmoveto␣show}def\n");
printf("/rs{moveto␣dup␣stringwidth␣pop␣neg␣0␣rmoveto␣show}def\n");
  /* Now draw the lines on the chart */
for (i ← 0; i < 22; i++) {
```

```
    printf("(%lg␣s)␣x␣-10␣cs\n", (double)(i * litvalue));
    printf("newpath␣x␣0␣moveto␣x␣bh␣%d␣mul␣lineto␣stroke␣/x␣x␣bw␣add␣def\n", framesize);
  }    /* Now write out the frequencies */
  for (i ← 0; i < framesize; i += 10) {
    printf("0␣%d␣bh␣mul␣.5␣bh␣mul␣add␣moveto␣-4␣0␣rlineto␣stroke\n", i);
    printf("(%lg)␣-7␣%d␣bh␣mul␣rs\n", (double)(i * SAMPLERATE)/(double)(framesize * 2), i);
  }    /* Now process the annotation labels. */
  printf("/Courier␣findfont␣7␣scalefont␣setfont\n");
  printf("/bw␣10␣72␣mul␣frames␣div␣def\n");
  printf("/y1␣-20␣def␣␣/y2␣-30␣def␣.2␣setlinewidth\n");
  printf("/lab␣{␣/x1␣exch␣def␣/x2␣exch␣def␣/str␣exch␣def\n");
  printf("␣newpath␣x1␣1␣add␣bw␣mul␣y1␣moveto␣0␣3␣rmoveto␣0␣-6␣rlineto\n");
  printf("␣0␣3␣rlineto␣x2␣bw␣mul␣y1␣lineto␣0␣3␣rmoveto␣0␣-6␣rlineto\n");
  printf("␣stroke␣newpath␣x1␣x2␣1␣add␣add␣bw␣mul␣2␣div␣y2␣moveto\n");
  printf("␣str␣stringwidth␣pop␣2␣div␣neg␣0␣rmoveto␣str␣show␣}␣def\n");
  printf("/cns␣{␣gsave␣-3␣0␣rmoveto␣0␣6␣rlineto␣dup␣stringwidth␣pop\n");
  printf("␣6␣add␣0␣rlineto␣0␣-6␣rlineto␣closepath␣1␣setgray␣fill\n");
  printf("␣grestore␣show␣}␣def\n");
  i ← tbeg ← 0;
  lab ← targetvals[0];
  while (i < nframes) {
    if (targetvals[i] ≠ lab) {
      printf("(%s)␣%d␣%d␣lab\n", tag_strings[lab], tbeg, i − 1);
      lab ← targetvals[i];
      tbeg ← i;
    }
    i++;
  }
  if (i − tbeg > 1) printf("(%s)␣%d␣%d␣lab\n", tag_strings[lab], tbeg, i − 1);
  printf("showpage\n");
}
```