```
void streamCluster( PStream* stream,
                    long kmin, long kmax, int dim,
                    long chunksize, long centersize, char* outfile )
{
  float* block = (float*)malloc( chunksize*dim*sizeof(float) );
  float* centerBlock = (float*)malloc(centersize*dim*sizeof(float) );
  long* centerIDs = (long*)malloc(centersize*dim*sizeof(long));

  if( block == NULL ) {
    fprintf(stderr,"not enough memory for a chunk!\n");
    exit(1);
  }

  Points points;
  points.dim = dim;
  points.num = chunksize;
  points.p = (Point *)malloc(chunksize*sizeof(Point));
  for( int i = 0; i < chunksize; i++ ) {
    points.p[i].coord = &block[i*dim];
  }

  Points centers;
  centers.dim = dim;
  centers.p = (Point *)malloc(centersize*sizeof(Point));
  centers.num = 0;

  for( int i = 0; i< centersize; i++ ) {
    centers.p[i].coord = &centerBlock[i*dim];
    centers.p[i].weight = 1.0;
  }

  long IDoffset = 0;
  long kfinal;
  while(1) {

    size_t numRead  = stream->read(block, dim, chunksize );
    fprintf(stderr,"read %d points\n",numRead);

    if( stream->ferror() || numRead < (unsigned int)chunksize && !stream->feof() ) {
      fprintf(stderr, "error reading data!\n");
      exit(1);
    }

    points.num = numRead;
    for( int i = 0; i < points.num; i++ ) {
      points.p[i].weight = 1.0;
    }

    switch_membership = (bool*)malloc(points.num*sizeof(bool));
    is_center = (bool*)calloc(points.num,sizeof(bool));
    center_table = (int*)malloc(points.num*sizeof(int));

    localSearch(&points,kmin, kmax,&kfinal);

    fprintf(stderr,"finish local search\n");
    contcenters(&points);
    if( kfinal + centers.num > centersize ) {
      //here we don't handle the situation where # of centers gets too large.
      fprintf(stderr,"oops! no more space for centers\n");
      exit(1);
  }

copycenters(&points, &centers, centerIDs, IDoffset);
IDoffset += numRead;

free(is_center);
free(switch_membership);
free(center_table);

if( stream->feof() ) {
   break;
```

```
    }
}
```