```c
static Header base;          /* empty list to get started */
static Header *freep = NULL; /* start of free list */

/* malloc: general-purpose storage allocator */
void *malloc(unsigned nbytes) {
  Header *p, *prevp;
  Header *morecore(unsigned);
  unsigned nunits;
  nunits = (nbytes + sizeof(Header) - 1) / sizeof(Header) + 1;
  if ((prevp = freep) == NULL) { /* no free list yet */
    base.s.ptr = freep = prevp = &base;
    base.s.size = 0;
  }
  for (p = prevp->s.ptr;; prevp = p, p = p->s.ptr) {
    if (p->s.size >= nunits) { /* big enough */
      if (p->s.size == nunits) /* exactly */
        prevp->s.ptr = p->s.ptr;
      else { /* allocate tail end */
        p->s.size -= nunits;
        p += p->s.size;
        p->s.size = nunits;
      }
      freep = prevp;
      return (void *)(p + 1);
    }
    if (p == freep) /* wrapped around free list */
      if ((p = morecore(nunits)) == NULL)
        return NULL; /* none left */
  }
}

static Header *morecore(unsigned nu) {
  char *cp, *sbrk(int);
  Header *up;
  if (nu < NALLOC)
    nu = NALLOC;
  cp = sbrk(nu * sizeof(Header));
  if (cp == (char *)-1) /* no space at all */
    return NULL;
  up = (Header *)cp;
  up->s.size = nu;
  free((void *)(up + 1));
  return freep;
}

/* free: put block ap in free list */
void free(void *ap) {
  Header *bp, *p;
  bp = (Header *)ap - 1; /* point to block header */
  for (p = freep; !(bp > p && bp < p->s.ptr); p = p->s.ptr)
    if (p >= p->s.ptr && (bp > p || bp < p->s.ptr))
      break;                         /* freed block at start or end of arena */
  if (bp + bp->s.size == p->s.ptr) { /* join to upper nbr */
    bp->s.size += p->s.ptr->s.size;
    bp->s.ptr = p->s.ptr->s.ptr;
  } else
    bp->s.ptr = p->s.ptr;
  if (p + p->s.size == bp) { /* join to lower nbr */
    p->s.size += bp->s.size;
    p->s.ptr = bp->s.ptr;
  } else
    p->s.ptr = bp;
  freep = p;
}
```