typedef long Align; /\* for alignment to long boundary \*/

union header { /\* block header \*/

struct {

union header \*ptr; /\* next block if on free list \*/

unsigned size; /\* size of this block \*/

} s;

Align x; /\* force alignment of blocks \*/

};

typedef union header Header;

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 = freeptr = prevptr = &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 \*/

}

}