LECTURE 25

LINE INPUT / OUTPUT (K&R, § 7.7)

input	fgets char *fgets(char *line, int maxline, FILE *fp)
	fgets is like getline() from a file
	reads the next input line (including '\n') from file fp into the char array line
	at most, maxline - 1 chars will be read
	a terminal '\0' is added at the end
	returns a pointer to line (or NULL if it reaches EOF)

output	fputs int fputs(char *line, FILE *fp)
	fputs writes line to fp
	returns 0 for successful write (or EOF if error occurs, e.g. disk fills up)
	in the event of an error, we can use perror to print out exact error cause to stderr

fgets and fputs implementations from K&R (page 165)

```
char *fgets(char *s, int n, FILE *iop)
{
  register int c;
  register char *cs;
  cs = s;
  while (--n >0 && (c = getc(iop)) != EOF))
    if ((*cs++ =c) == '\n')
      break;
  *cs = '\0';
   return (c == EOF && cs == s) ? NULL : s;
}
```

```
int fputs(char *s, FILE *iop)
{
  int c;
  while (c = *s++)
    putc(c, iop);
  return ferror(iop) ? EOF : 0;
}
```

MISCELLANEOUS FUNCTIONS (K&R, § 7.8)

STRING OPERATIONS

We've used many of these throughout the semester.
s, t are char * (character pointers)
c, n are ints

strcat(s,t)	concatenate t to end of s
strncat(s,t,n)	concatenate n characters of t to end of s
strcmp(s,t)	return negative (s < t) zero (s == t) positive (s > t)
strncmp(s,t,n)	same as strcmp but only in first n characters
strcpy(s,t)	copy t to s
strncpy(s,t,n)	copy at most n characters of t to s
strlen(s)	return length of s
strchr(s,c)	return pointer to first c in s, or NULL if not present
strrchr(s,c)	return pointer to last c in s, or NULL if not present

CLASS TESTING AND CONVERSION

We've used at least one of these.
c is an int that can be represented as an unsigned char or EOF. These functions return int.

isalpha(c)	non-zero if c is alphabetic, 0 if not
isupper(c)	non-zero if c is upper case, 0 if not
islower(c)	non-zero if c is lower case, 0 if not
isdigit(c)	non-zero if c is digit, 0 if not
isalnum(c)	non-zero if isalpha(c) or isdigit(c), 0 if not
isspace(c)	non-zero if c is blank, tab, newline, return, formfeed, vertical tab
toupper(c)	return c converted to upper case
tolower(c)	return c converted to lower case

MISCELLANEOUS

int ungetc(int c, FILE *fp)

Like the ungetch function we used in the calcit homework, but limited to one char at a time.

If successful, puts one character back to a stream, making it available for the next read operation, and returns the char it put back.

If it fails, returns EOF and stream is unchanged.

int system(char *s)

Passes the string s to the environment for execution, then resumes execution of the current program.

Examples:

```
int a, b;
char command[MAXCMD];
sprintf(command, "prog %d %d > prog.out", a, b);
system(command);
```

Allows you to run a program compiled as prog that takes command line input with inputs a and b. You can then access the output file (prog.out) in the calling program.

The return value is system-dependent. In UNIX, returns the value returned by exit.

Storage Management

void *malloc(size_t n)	We've used this a lot. Returns a void pointer to n bytes of uninitialized storage (or NULL if it fails)
void*calloc(size_t n, size_t size)	Returns a pointer to enough space for an array of n objects of size size bytes.

We must cast the returned pointer to the correct type of pointer for what we requested.

When we're done with the requested memory, we call free(pointer) to free it.

MATHEMATICAL FUNCTIONS

An incomplete listing of the mathematical functions available in <math.h>:

sin(x)	sine of x, x in radians
cos(x)	cosine of x, x in radians
atan2(y,x)	arctangent of y/x, in radians
exp(x)	exponential function e^x
log(x)	natural (base *e*) logarithm of x (x > 0)
log10(x)	common (base 10) logarithm of x (x > 0)
pow(x,y)	x^y
sqrt(x)	square root of x (x > 0)
fabs(x)	absolute value of x

NOTE: Remember that you must compile with the math library flag when using the math library:

```
gcc source.c -lm
```

(the -lm has to be at the end of the line)

RANDOM NUMBER GENERATION

int rand(void)

rand() returns a pseudo-random number in the range of 0 to RAND_MAX.

RAND_MAX is defined in <stdlib.h>. Its value is implementation-dependent, but guaranteed >= 32,767.

Example:
one way to produce random floating point numbers
0 <= random number < 1

```
#define frand() ((double) rand() / (RAND_MAX + 1.0))
```

void srand(unsigned int seed)

srand sets the seed value for rand. Often the seed value is based on the system clock (the number of seconds elapsed since 00:00:00 UTC, 1/1/1970).

QSORT

C's <stdlib.h> contains a quicksort implementation.

```
void qsort(void *base,
           size_t nitems,
           size_t size,
           int (*comp)(const void *,
           const void*)
          )
```

base	a pointer to the first element in the array to be sorted
nitems	the number of elements in the array to be sorted
size	the size (in bytes) of each element in the array
comp	function pointer to the compare used to sort

qsort has no idea what type of data it is sorting, this is why we must provide the size of each element.

So, if we have a basic compare function for ints:

```
int intcompare(int *i, int *j)
{
  return (*i - *j);
}
```

We can use qsort like this:

```
main ()
{
  int i;
  int a [10] = {8, 2, 9, 6, 5, 1, 3, 7, 4, 0};
  qsort(a,  // we want to sort array a
        10, // there are 10 elements in a
        sizeof(int), // each element has size 4 bytes
        // and we want to use intcompare
        (int (*) (void *, void *)) intcompare
        );
  for (i = 0; i < 10; i++)
    printf("%d, ", a[i]);
}
```

BSEARCH

C's <stdlib.h> contains a binary search implementation.

```
void *bsearch(const void *key,
              const void *base,
              size_t nitems,
              size_t size,
              int (*comp)(const void *, const void *)
             )
```

key	pointer to the object we're searching for
base	pointer to the first item in the array we are searching in
nitems	the number of elements in base
size	the size of each element in base
comp	function pointer to the compare we're using

bsearch returns a pointer to an entry in the array that matches the key (if the key is not found, a NULL pointer is returned).