PRECEDENCE OF ALL C OPERATORS
(Operators between horizontal lines have the same precedence)

Description	*Operator*	*Associates from the*	*Precedence*
Function expr	`()`	left	High (Evaluated first)
Array expr	`[]`		
`struct` indirection	`->`		
`struct` member	`.`		
Incr/decr	`++ --`	right	
One's complement	`~`		
Unary not	`!`		
Address	`&`		
Dereference	`*`		
Cast	`( type )`		
Unary plus	`+`		
Unary minus	`-`		
Size in bytes	`sizeof`		
Multiplication	`*`	left	
Division	`/`		
Modulus	`%`		
Addition	`+`		
Subtraction	`-`		
Shift left	`<<`		
Shift right	`>>`		
Less than	`<`		
Less than or equal	`<=`		
Greater than	`>`		
Greater than or equal	`>=`		
Equal	`==`		
Not equal	`!=`		
Bitwise and	`&`		
Bitwise exclusive or	`^`		
Bitwise inclusive or	`|`		
Logical and	`&&`		
Logical or	`||`		
Conditional	`? :`	right	
Assignment	`= %= += -= *= /= >>= <<= &= ^= |=`		
Comma	`,`	left	(Evaluated last) Low

CODES FOR `scanf`, `fscanf`, AND `sscanf`

Code	*Interpretation*	*Example of Input*	*Corresponding Argument Must Be*
`c`	a character	`p`	address of `char`
`s`	a character string	`pie`	address of `char`
`d`	decimal integer converted to `int`	`27649`	address of `int`
`hd`	decimal integer converted to `short`	`-6942`	address of `short`
`ld`	decimal integer converted to `long`	`2964775`	address of `long`
`o`	octal integer converted to `int`	`65777`	address of `int`
`ho`	octal integer converted to `short`	`5436`	address of `short`
`lo`	octal integer converted to `long`	`-7255547`	address of `long`
`x, X`	hexadecimal integer converted to `int`	`-6bff`	address of `int`
`hx, hX`	hexadecimal integer converted to `short`	`B2E`	address of `short`
`lx, lX`	hexadecimal integer converted to `long`	`2d3d27`	address of `long`
`i`	integer [decimal, octal (leading `0`), or hex (leading `0x` or `0X`)] converted to `int`	`065702`	address of `int`
`hi`	integer converted to `short`	`0xB2E`	address of `short`
`li`	integer converted to `long`	`2964775`	address of `long`
`u`	`unsigned int`	`62548`	address of `unsigned int`
`hu`	`unsigned short`	`46927`	address of `unsigned short`
`lu`	`unsigned long`	`3694207846`	address of `unsigned long`
`e, f, g, E, G`	floating-point number converted to `float`	`4.13986e+03`	address of `float`
`le, lf, lg, lE, lG`	floating-point number converted to `double`	`3.141592654`	address of `double`
`Le, Lf, Lg, Le, LG`	floating-point number converted to `long double`	`3.14159265358979323846`	address of `long double`
`p`	address	(implementation dependent)	address of a pointer to `void`
`n`	number of characters read so far (store in `int`)	(none)	address of `int`
`hn`	number of characters read so far (store in `short`)	(none)	address of `short`
`ln`	number of characters read so far (store in `long`)	(none)	address of `long`
`[chars]`	string that includes only *chars*	`edit` (if the conversion is `[abcdefghit]`)	address of `char`
`[^chars]`	string that excludes only *chars*	`edit` (if the conversion is `[^xyz]`)	address of `char`
`%`	match the character `%` in the input	`%`	(none)

Math Functions

`abs`	Absolute value of an `int`	`floor`	Floor
`acos`	Arccosine	`labs`	Absolute value of a `long`
`asin`	Arcsine	`log`	$\log_e x$
`atan`	Arctangent	`log10`	$\log_{10} x$
`atof`	Convert string to `double`	`pow`	x^y
`atoi`	Convert string to `int`	`rand`	Generate a random integer
`atol`	Convert string to `long`	`sin`	Sine
`ceil`	Ceiling	`sinh`	Hyperbolic sine
`cos`	Cosine	`sqrt`	Square root
`cosh`	Hyperbolic cosine	`srand`	Seed the random number generator
`exp`	e^x	`tan`	Tangent
`fabs`	Absolute value of a `double`	`tanh`	Hyperbolic tangent

Input/Output Functions

`fclose`	Close a file	`getc`	Read a character
`feof`	Check end-of-file	`getch`	Read a character
`fgetc`	Read a character	`getchar`	Read a character
`fgets`	Read a string	`gets`	Read a string
`fopen`	Open a file	`printf`	Write formatted output
`fprintf`	Write formatted output	`putc`	Write a character
`fputc`	Write a character	`putchar`	Write a character
`fputs`	Write a string	`puts`	Write a string
`fread`	Read several items	`rewind`	Move to beginning of file
`fscanf`	Read formatted input	`scanf`	Read formatted input
`fseek`	Move within a file	`sprintf`	Write formatted output
`ftell`	Find position within a file	`sscanf`	Read formatted input
`fwrite`	Write several items	`ungetc`	Return a character to a buffer

Miscellaneous Functions

`bsearch`	Binary search
`clearerr`	Clear end-of-file and error indicators
`difftime`	Compute difference between times
`exit`	Terminate program
`longjmp`	Restore environment
†`lsearch`	Linear search
`qsort`	Quicksort
`setjmp`	Set jump
`signal`	Invoke a function to handle a signal
`system`	Execute a command
`time`	Find time

†Function not specified by standard.

Continued on following page.

String Functions

`memchr`	Find leftmost character in object
`memcmp`	Compare objects
`memcpy`	Copy object
`memmove`	Copy object
`strcat`	Concatenate strings
`strchr`	Find leftmost character in string
`strcmp`	Compare strings
`strcpy`	Copy string
`strcspn`	Complement of span
`strlen`	Length of string
`strncat`	Concatenate strings
`strncmp`	Compare strings
`strncpy`	Copy string
`strpbrk`	Find break character
`strrchr`	Find rightmost character in string
`strspn`	Span
`strstr`	Find substring

Type and Conversion Functions

`atof`	Convert string to `double`
`atoi`	Convert string to `int`
`atol`	Convert string to `long`
`isalnum`	Alphanumeric?
`isalpha`	Alphabetic character?
`iscntrl`	Control character?
`isdigit`	Decimal digit?
`isgraph`	Nonblank, printable character?
`islower`	Lowercase character?
`isprint`	Printable character?
`ispunct`	Punctuation character?
`isspace`	Space character?
`isupper`	Uppercase character?
`isxdigit`	Hexadecimal character?
`tolower`	Convert from uppercase to lowercase
`toupper`	Convert from lowercase to uppercase

Nonstandard, Borland C++ Graphics Functions

`bar`	Draw and fill rectangle
`circle`	Draw circle
`closegraph`	Close graphics system
`getmaxx`	Get maximum x-coordinate of screen
`getmaxy`	Get maximum y-coordinate of screen
`grapherrormsg`	Give error message
`graphresult`	Describe last graphics error
`initgraph`	Initialize graphics system
`line`	Draw line (absolute coordinates)
`linerel`	Draw line (relative coordinates)
`moveto`	Change current pixel position
`outtextxy`	Write string

Memory Allocation Functions

`calloc`	Allocate storage
`free`	Free storage
`malloc`	Allocate storage

- double sin(double x);
- double cos(double x);
- double tan(double x);
- double exp(double x); /* computes e^x */
- double log(double x); /* computes the natural logarithm of x */
- double log10(double x); /* computes the common (base ten) logarithm of x */
- double fabs(double x); /* absolute value of x */
- double pow(double x, double y); /* computes x^y */

- <stdio.h>
  - int fclose(FILE *stream);
  - int feof(FILE *stream);
  - FILE *fopen(const char *filename, const char *mode);
  - int fprintf(FILE *stream, const char *format, ...);
  - int fscanf(FILE *stream, const char *format, ...);
  - int getchar(void);
  - int printf(const char *format, ...);
  - int scanf(const char *format, ...);
- <stdlib.h>
  - RAND_MAX
  - int rand(void);
  - void srand(unsigned int seed);
- <conio.h>
  - int kbhit(void); /* returns 0 if no keystrokes are currently in buffer */
  - void gotoxy(int x, int y); /* position cursor at row y, column x. 1,1 is upper left */
  - int getch(void); /* get one character from keyboard buffer (without carriage return) */

IEEE Floating Point Representation

- N = number of bits in data
- Mn = number of bits in mantissa
- En = number of bits in exponent
- s = sign bit
- ordering within data value: [s][En][Mn]
- M is the unsigned value of the bits Mn
- E is the unsigned value of the bits En

type	bytes	N	Mn	En
float	4	32	23	8
double	8	64	52	11
long double	10	80	64	15

if (E equals 0):

- The exponent is: $e = -(2^{(En-1)} - 2)$
- The mantissa is: $m = M/s^{Mn}$

else

- The exponent is: $e = E - (2^{(En-1)} - 1)$
- The mantissa is: $m = 1 + M/s^{Mn}$

The final value is

- value = $(-1)^s * (m) * (2)^e$

APPENDIX A

ASCII AND EBCDIC TABLES

TABLE A.1 ASCII Codes

Decimal	*Hexadecimal*	*Octal*	*Standard Function*
0	00	000	NUL (Null)
1	01	001	SOH (Start of heading)
2	02	002	STX (Start of text)
3	03	003	ETX (End of text)
4	04	004	EOT (End of transmission)
5	05	005	ENQ (Enquiry)
6	06	006	ACK (Acknowledge)
7	07	007	BEL (Ring bell)
8	08	010	BS (Backspace)
9	09	011	HT (Horizontal tab)
10	0A	012	LF (Line feed)
11	0B	013	VT (Vertical tab)
12	0C	014	FF (Form feed)
13	0D	015	CR (Carriage return)
14	0E	016	SO (Shift out)
15	0F	017	SI (Shift in)
16	10	020	DLE (Data link escape)
17	11	021	DC1 (Device control 1)
18	12	022	DC2 (Device control 2)
19	13	023	DC3 (Device control 3)
20	14	024	DC4 (Device control 4)
21	15	025	NAK (Negative acknowledge)
22	16	026	SYN (Synchronous idle)
23	17	027	ETB (End of transmission block)
24	18	030	CAN (Cancel)
25	19	031	EM (End of medium)
26	1A	032	SUB (Substitute)

TABLE A.1 ASCII Codes (*continued*)

Decimal	Hexadecimal	Octal	Standard Function
27	1B	033	ESC (Escape)
28	1C	034	FS (File separator)
29	1D	035	GS (Group separator)
30	1E	036	RS (Record separator)
31	1F	037	US (Unit separator)
32	20	040	SP (Space)
33	21	041	!
34	22	042	"
35	23	043	#
36	24	044	$
37	25	045	%
38	26	046	&
39	27	047	' (Single quote)
40	28	050	(
41	29	051	)
42	2A	052	*
43	2B	053	+
44	2C	054	, (Comma)
45	2D	055	- (Hyphen)
46	2E	056	.
47	2F	057	/
48	30	060	0
49	31	061	1
50	32	062	2
51	33	063	3
52	34	064	4
53	35	065	5
54	36	066	6
55	37	067	7
56	38	070	8
57	39	071	9
58	3A	072	:
59	3B	073	;
60	3C	074	<
61	3D	075	=
62	3E	076	>
63	3F	077	?
64	40	100	@
65	41	101	A
66	42	102	B
67	43	103	C
68	44	104	D
69	45	105	E
70	46	106	F

TABLE A.1 ASCII Codes (*continued*)

Decimal	Hexadecimal	Octal	Standard Function
71	47	107	G
72	48	110	H
73	49	111	I
74	4A	112	J
75	4B	113	K
76	4C	114	L
77	4D	115	M
78	4E	116	N
79	4F	117	O
80	50	120	P
81	51	121	Q
82	52	122	R
83	53	123	S
84	54	124	T
85	55	125	U
86	56	126	V
87	57	127	W
88	58	130	X
89	59	131	Y
90	5A	132	Z
91	5B	133	[
92	5C	134	\
93	5D	135	]
94	5E	136	^
95	5F	137	_ (Underscore)
96	60	140	` (Grave accent)
97	61	141	a
98	62	142	b
99	63	143	c
100	64	144	d
101	65	145	e
102	66	146	f
103	67	147	g
104	68	150	h
105	69	151	i
106	6A	152	j
107	6B	153	k
108	6C	154	l
109	6D	155	m
110	6E	156	n
111	6F	157	o
112	70	160	p
113	71	161	q
114	72	162	r

`c`	a character	`p`	`char, short, int`
`s`	a character string	`pie`	address of `char`
`d, i`	integer written in signed decimal notation	`-999`	`char, short, int`
`ld, li`	`long` written in signed decimal notation	`2964775`	`long`
`o`	integer written in unsigned octal notation	`7031`	`char, short, int`
`lo`	`long` written in unsigned octal notation	`7255547`	`long`
`x`	integer written in unsigned hexadecimal notation (using `a-f`)	`fe4`	`char, short, int`
`X`	integer written in unsigned hexadecimal notation (using `A-F`)	`FE4`	`char, short, int`
`lx`	`long` written in unsigned hexadecimal notation (using `a-f`)	`2d3d27`	`long`
`lX`	`long` written in unsigned hexadecimal notation (using `A-F`)	`2D3D27`	`long`
`u`	`unsigned` decimal	`287092`	`unsigned char, unsigned short, unsigned int`
`lu`	`unsigned long`	`3775287092`	`unsigned long`
`e`	`float` or `double` written in the form `m.nnnnnne±xx`	`3.141590e+03`	`float, double`
`E`	`float` or `double` written in the form `m.nnnnnnE±xx`	`3.141590E+03`	`float, double`
`f`	`float` or `double` written in the form `m.nnnnnn`	`2.718282`	`float, double`
`g`	number written in the `d`, `e`, or `f` format, whichever is shortest	`33.900000`	`float, double`
`G`	number written in the `d`, `E`, or `f` format, whichever is shortest	`33.900000`	`float, double`
`Le`	`long double` written in the form `m.nnnnnne±xx`	`3.141590423e+03`	`long double`
`LE`	`long double` written in the form `m.nnnnnnE±xx`	`3.141590423E+03`	`long double`
`Lf`	`long double` written in the form `m.nnnnnn`	`3.141592654`	`long double`
`Lg`	number written in the `d`, `Le`, or `Lf` format, whichever is shortest	`43.9000001`	`long double`
`LG`	number written in the `d`, `LE`, or `Lf` format, whichever is shortest	`43.9000001`	`long double`
`p`	address	(implementation dependent)	pointer to `void`
`n`	number of characters written so far is stored in `int` whose address is passed	(none)	address of `int`
`hn`	number of characters written so far is stored in `short` whose address is passed	(none)	address of `short`
`ln`	number of characters written so far is stored in `long` whose address is passed	(none)	address of `long`
`%`	write the character `%`	`%`	(none)

FIGURE 9.4 Codes for `printf`, `fprintf`, and `sprintf`.

`-`	Left-justify in field
`+`	Begin a signed number with `+` or `-`
blank	If a signed number does not begin with `+` or `-`, add a space before printing it
`#`	For `o` conversion, begin number with `0` (zero)
	For `x` conversion, begin number with `0x`
	For `X` conversion, begin number with `0X`
	For `e`, `E`, `f`, `g`, and `G` conversions, use a decimal point
	For `g` and `G` conversions, do not remove trailing zeros
`0` (zero)	Pad a number on the left with zeros

FIGURE 9.3 Flags used in the `printf` format string.

Selected function prototypes and constant from the standard libraries. You shouldn't need any functions beyond those given here. If you feel you do need one, ask the proctor who will write the prototype on the board.

- <math.h>
  - double sin(double x);
  - double cos(double x);
  - double tan(double x);
  - double exp(double x); /* computes e^x */
  - double log(double x); /* computes the natural logarithm of x */
  - double log10(double x); /* computes the common (base ten) logarithm of x */
  - double fabs(double x); /* absolute value of x */
  - double pow(double x, double y); /* computes x^y */
- <stdio.h>
  - int fclose(FILE *stream);
  - int feof(FILE *stream);
  - FILE *fopen(const char *filename, const char *mode);
  - int fprintf(FILE *stream, const char *format, ...);
  - int fscanf(FILE *stream, const char *format, ...);
  - int getchar(void);
  - int printf(const char *format, ...);
  - int scanf(const char *format, ...);
- <stdlib.h>
  - RAND_MAX
  - int rand(void);
  - void srand(unsigned int seed);
  - void *malloc(size_t size);
  - void free(void *block);
- <conio.h>
  - int kbhit(void); /* returns 0 if no keystrokes are currently in buffer */
  - void gotoxy(int x, int y); /* position cursor at row y, column x. 1,1 is upper left */
  - int getch(void); /* get one character from keyboard buffer (without carriage return) */

TWO'S COMPLEMENT representation for negative integers

-(n) = (~n) + 1 (for n positive OR negative)

- Invert all bits in the integer representation.
- Add 1

HEXADECIMAL TABLE

DEC	HEX	BIN	DEC/16	DEC	HEX	BIN	DEC/16
0	0	0000	0.0000	8	8	1000	0.5000
1	1	0001	0.0625	9	9	1001	0.5625
2	2	0010	0.1250	10	A	1010	0.6250
3	3	0011	0.1875	11	B	1011	0.6875
4	4	0100	0.2500.	12	C	1100	0.7500.
5	5	0101	0.3125	13	D	1101	0.8125
6	6	0110	0.3750	14	E	1110	0.8750
7	7	0111	0.4375	15	F	1111	0.9375