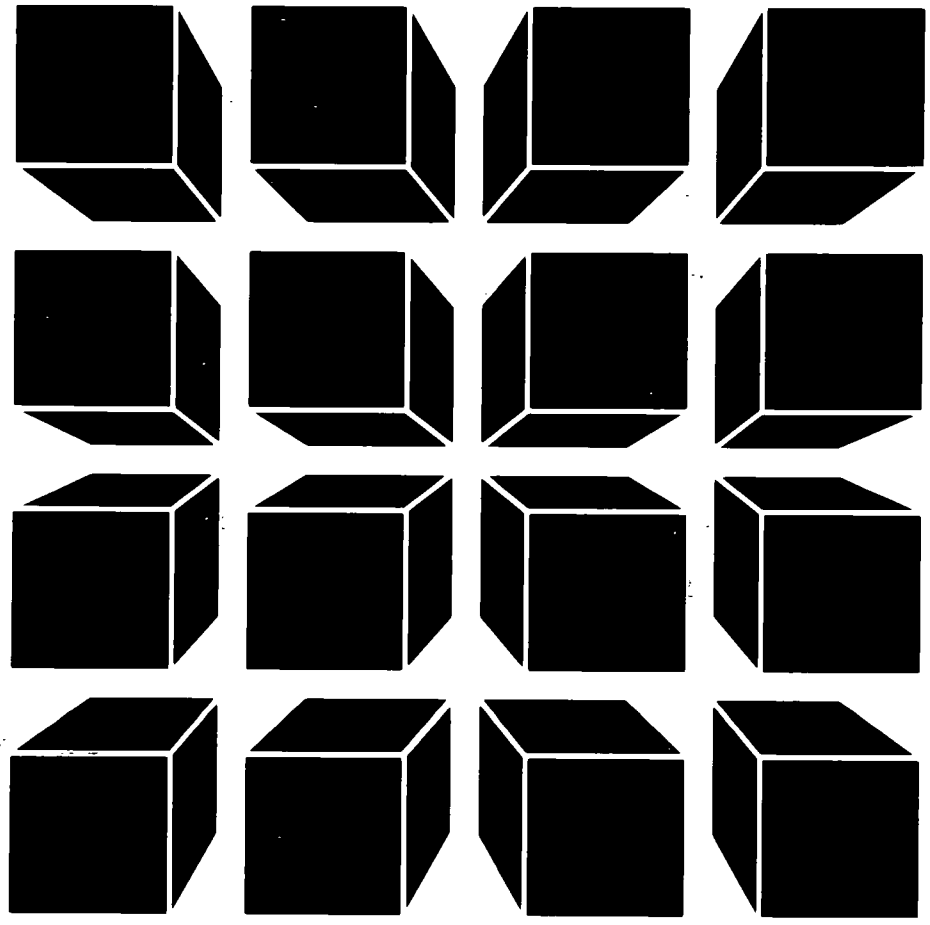
AT&T VIDEOTAPE LIBRARY

C LANGUAGE FOR PROGRAMMERS



WORKBOOK VOL I

AT&T

WORKBOOK VOLUME I

CONTENTS

VOLUME 7—The C Preprocessor and Program Organization

WORKBOOK VOLUME II

CONTENTS

VOLUME 8—Introduction to Pointers

VOLUME 9—Structures and Unions

AT&T COMPUTER SYSTEMS
EDUCATION PROGRAM
INTRODUCTION TO STUDENT WORKBOOK

To the Student,

Welcome to the AT&T Videotape Library. We hope you will enjoy using the videotape lessons and this workbook. You will find the material both practical and easy to follow. You will be using what you learn in your daily work almost immediately.

If you haven't taken a class by videotape before, you are in for an exciting and fun experience, because this medium gives you control over the pace of your learning. Each videotape lesson is divided into segments which are clearly marked with colored panels, so you can find them easily while using fast forward or rewind. When you encounter new terms or concepts, you can immediately review and reinforce them by using the videotape player's rewind feature. If you find some material that is already familiar to you, simply use the player's fast forward button to jump ahead to the next segment. You can even stop the tape completely to take a break. The lesson will pick up right where you left off, and you won't miss a word.

This course, on the **C Language for Programmers**, will provide you with a working knowledge of C, and enable you to write and implement powerful and versatile C programs. The course examines all C operators, program flow control constructs, arrays, structures, and file I/O.

This workbook has been prepared as a support tool for this videotape course. Each volume in the workbook follows one lesson. The workbook contains summaries and reviews of what you learned in each videotape lesson, as well as supplemental examples, and short exercises to be performed during the tape presentation. The instructor will ask you to stop the tape and work the appropriate exercises. These may be done with pencil and paper.

VOLUME 1

Getting Started — Part 1

C LANGUAGE KEYWORDS

auto	enum	short
break	extern	sizeof
case	float	static
char	for	struct
continue	goto	switch
default	if	typedef
do	int	union
double	long	unsigned
else	register	while
entry	return	void

- An identifier name may not be a keyword.

CONSTANTS

- Integer

Decimal	55	100	255
Octal	067	0144	0377
Hex	0x37	0x67	0Xff
Explicit Long	55L	100l	255L

- Floating Point

```
5.32   5.   .75
.240E100   25e-3
```

- Character

Printable	'a' 'B' '+' '5'	
Special	'\n'	newline
	'\t'	horizontal tab
	'\b'	backspace
	'\r'	carriage return
	'\f'	form feed
	'\\'	backslash
	'\''	single quote
	'\v'	vertical tab
	'\0'	NULL
Bit pattern	'\007' '\012' '\377'	

- String Literal - delimited by '\0'

```
"Please enter your first name"
"Annual Report"
```

VOLUME 1 EXERCISE

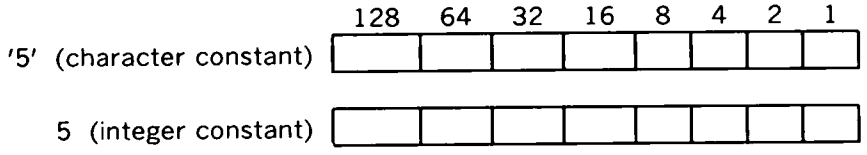
Valid or Invalid Identifier Names?	
x	num_parts
6x	part5
VAL	table.index
PrevAmt	$pay
employee_name	employee_pay

==========

Interpret these four bytes as ASCII characters:

Binary	Oct	Hex	Dec	Character
01000011	0103	0x43	67	
01101111	0157	0x6f	111	
01100100	0144	0x64	100	
01100101	0145	0x65	101	

What is the bit pattern (binary code) for



VOLUME 2

Getting Started — Part 2

OPERATOR PRECEDENCE

TYPE	OPERATOR	ASSOCIATIVE
PRIMARY	() [] -> .	L TO R
UNARY	! ~ - ++ -- (TYPE) * & sizeof	R TO L
MULTIPLICATIVE	* / %	L TO R
ADDITIVE	+ -	L TO R
SHIFT	<< >>	L TO R
RELATIONAL	< <= > >=	L TO R
EQUALITY	== !=	L TO R
BITWISE	&	L TO R
BITWISE	^	L TO R
BITWISE	|	L TO R
LOGICAL	&&	L TO R
LOGICAL	||	L TO R
CONDITIONAL	?:	R TO L
ASSIGNMENT	= op=	R TO L
COMMA	,	L TO R

- **Parentheses - clarity, change precedence**

OPERATOR - SUMMARY

Additive	+	add
	-	subtract
Multiplicative	*	multiply
	/	divide
	%	remainder
Unary Negation	-	e.g., -x
Relational†	<	less than
	<=	less than or equal to
	>	greater than
	>=	greater than or equal to
	==	equal to
	!=	not equal to
Logical†	&&	and
	||	or
	• left to right evaluation	
	• execution stops when truth value determined	
Logical Negation†	!	not
Assignment	=	assign
	op=	x *op*= y is equivalent to x = x *op* y

†Value of expression is 0 (false) or 1 (true)
(Relational, Logical and Logical Negation)
☞ **0 is false, nonzero is true** ☜

VOLUME 2 EXERCISE
EXPRESSIONS and STATEMENTS

What's wrong here?

```
a = 0;
while ( a < 10 )
        a = a + 1;
        b = b + a;
```

```
a = 0;
while ( a < 10 ) ;
        {
        a = a + 1;
        b = b + a;
        }
```

```
/* Discard new-line characters */
input = getchar();
while (input = '\n') {
        input = getchar();
        c = c + 1;
        }
```

VOLUME 2 EXERCISE - ANSWERS

What's wrong here?

1. Indenting does not take the place of { }.

2. Body of loop is the null statement.

3. Confusing = with ==.

CHARACTER I/O

NAME	getchar	putchar
SYNOPSIS	#include <stdio.h> int getchar()	#include <stdio.h> int putchar(c) int c;
DESCRIPTION	Reads and returns next character from *standard input.*	Writes character *c* to *standard output.*
DIAGNOSTICS	Returns EOF* at end-of-file or error.	Returns value written if successful, else EOF*.

==========

EXAMPLES

```
 1 #include     <stdio.h>
 2 main()
 3 {            int c;
 4
 5              c = getchar();          /*  read character */
 6              putchar(c);             /* write character */
 7              putchar('M');
 8              putchar('\t');
 9              putchar('\007');
10 }
```

* EOF is defined in stdio.h; typical value is -1.

VOLUME 2 SUMMARY

Program format:

```
main()
{
   declarations
   statements
}
```

Identifiers: alphas, digits, _

Basic Data Types:

```
char
int  (short,long,unsigned)
float
double
```

Operators:

```
+ -
* / %
< <= > >=
== != !
= op=
```

expression - has a value

constant
identifier
expression(s) and operator(s)
(expression)

statement - action item

expression;

```
{
  optional declarations
  statement(s)
}
```

if, while,for, etc.

VOLUME 2 SUMMARY

Error messages are generated if:

1. File name doesn't end with ".c"
2. A semicolon is missing
3. An undeclared variable is used
4. The second " in printf is missing

and many others ...

B. Arithmetic Calculations

1. A given IRA has an interest rate of 7 percent compounded yearly.

   a. If you put $2000 into the IRA one year, how much is the IRA worth after 20 years?

   b. How much is the IRA worth if you put $2000 into it every year for 20 years?

2. Assume the following is true:

MODEL	MILES PER GALLON
Guzzler	8
Average	33
Lean Machine	45

   a. If gas costs an average of $.96 per gallon and someone drives 16,000 miles per year, how much would be spent on gas per year for each of the three cars?

   b. Over a six-year period, what is the dollar savings on gas for the "Lean Machine" over the "Average"?
   over the "Guzzler"?

```
/* 2A.3.c
 * Copies standard input to standard output, converting uppercase to
 * lowercase.  (Assumes standard input is ASCII characters.) */

#include    <stdio.h>

main()
{
     int  c, offset;

     /* Find difference between ASCII codes for 'a' and 'A' */
     offset = 'a' - 'A';
     while (( c = getchar()) != EOF) { /* Reading until end-of-file */
          if ( c >= 'A' && c <= 'Z')
               c = c + offset;
          putchar(c);
          }
 }

/* 2A.4.c
 * Reads from standard input, writes one word per line.  */

#include     <stdio.h>

main()
{
     int  c;

     while ( (c = getchar()) != EOF) {
          if (c == ' ' || c == '\t')              /* end of word reached */
               putchar('\n');
          else
               putchar(c);
          }
}
```

VOLUME 2 LAB - SOLUTIONS

```
/* 2B.2a.c
  Finds dollars spent on gas per year for three different
  cars, assuming gas costs an average of $.96 per gallon,
  16,000 miles are driven per year, and the cars get the
  following mileages per gallon:

          Guzzler - 8 mpg - $1,920.00
          Average - 33 mpg - $465.45
          Lean Machine - 45 mpg - $341.33
*/
#include     <stdio.h>
main()
{
     int  guzzler = 8, average = 33, lean = 45;

     printf("Dollars per year:\n");
     printf("Guzzler:  $%.2f\n", (16000.0 / guzzler) * .96);
     printf("Average:  $%.2f\n", (16000.0 / average) * .96);
     printf("Lean Machine:  $%.2f\n", (16000.0 / lean) * .96);
}
```

```
/* 2B.2b.c
  Finds dollars saved over a 6-year period if the "Lean Machine"
  is driven.
  Lean Machine over Guzzler: $9,472.00
  Lean Machine over Average: $ 744.73
*/
#include     <stdio.h>
main()
{
     int  guzzler = 8, average = 33, lean = 45;

     printf("Dollars saved over a 6-year period:\n");
     printf("Lean Machine over Guzzler: $%.2f\n",
          6 * (16,000.0 / guzzler - 16,000.0 / lean) * .96);
     printf("Lean Machine over Average: $%.2f\n",
          6 * (16,000.0 / average - 16,000.0 / lean) * .96);
}
```

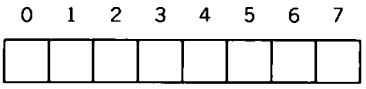
VOLUME 3

Arrays and Control Flow

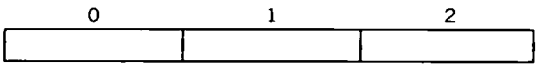
ARRAYS

- Collection of Identically-Typed Objects
- Single or Multi-dimensional
- Sequential Storage
- Index: 0 to (size - 1)

char id[8];



float price[3];



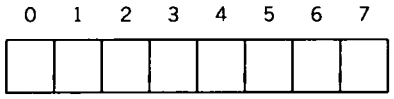
int table[4][3];

	col0	*col1*	*col2*
row0	0 0	0 1	0 2
row1	1 0	1 1	1 2
row2	2 0	2 1	2 2
row3	3 0	3 1	3 2

ARRAY ELEMENT REFERENCE

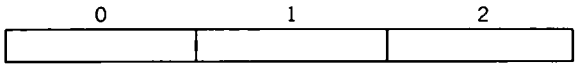
array_name[*integer expression*]

char id[8];




```
id[0] = 'j';
id[1] = id[0];
```

float price[3];




```
num = 2;
price[num] = price[num - 1] * 2;
```

int table[4][3];

	col0	*col1*	*col2*
row0	0 0	0 1	0 2
row1	1 0	1 1	1 2
row2	2 0	2 1	2 2
row3	3 0	3 1	3 2

```
printf("%d\n",table[2][1]);
```

COPYING ARRAYS

☞ Arrays must be copied element by element. ☜

It is illegal to use *just* an array name on the left side of an assignment statement. An array name represents the address of where the array is stored. This address cannot be changed; it is not an *lvalue*.

```
int     prev[20], current[20], i;
```

Incorrect:

```
        prev = current;   /* Compile error */
```

Correct:

```
        i = 0;
        while (i < 20) {
                prev[i] = current[i];
                i = i + 1;
                }
```

VOLUME 3 EXERCISE
ARRAYS INCREMENT (DECREMENT OPERATOR)

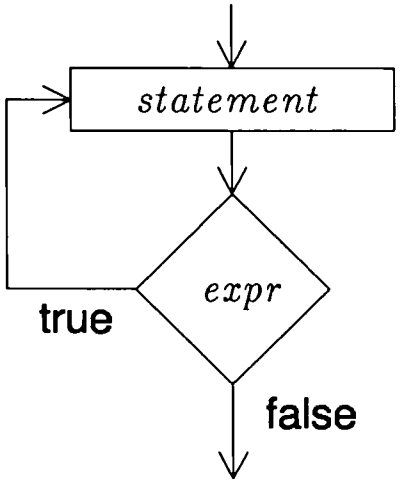
3. What are the values of x and y after each set of statements?

x = 3;
 	
y = ++x;
	x = 3;
	
	y = x++;
		x = 3;
		
		y = --x;
			x = 3;
			
			y = x--;

y _____ x _____	y _____ x _____	y _____ x _____	y _____ x _____

DO-WHILE LOOP

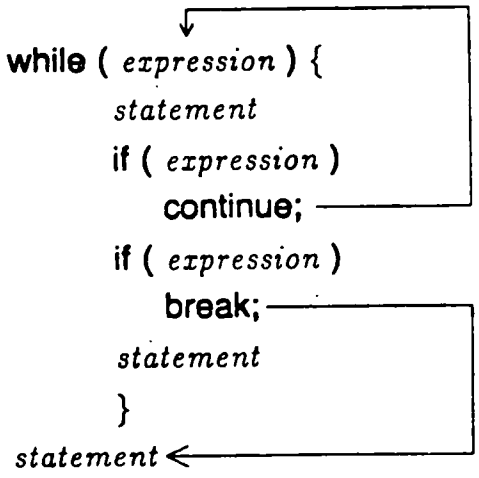
do
statement
while (*expression***);**




```
1   #include        <stdio.h>
2   main()   /* Do print menu while choice is not 'q' */
3   {
4      char   choice;
5      do {   while (getchar() != '\n')
6                  ;   /* Clear line */
7             printf("\tMAIN MENU\n");
8             printf("d - data entry\n");
9             printf("r - report\n");
10            printf("q - quit\n");
11               code for getting menu choice here
12            if (choice == 'd')
13               code for getting data entry here
14            else if (choice == 'r')
15               code for printing report here
16            else if (choice != 'q')
17                  printf("Illegal choice.Try again.\n");
18            }
19     while (choice != 'q');
20  }
```

BREAK and CONTINUE STATEMENTS

break	Causes immediate exit from innermost loop (while, do-while, for) or switch*
continue	Causes next iteration of innermost loop (while, do-while, for) to begin.



* Covered later in this unit

SWITCH STATEMENT

- multi-way decision maker
- alternative to nested if-else when comparing expression to various constants
- cases act as labels
- default case optional; breaks optional

```
switch (expression) {
        case constant:   statement(s)
        case constant:   statement(s)
        default:statement(s)
        }
```

```
/* switch example */

switch (num)  {
  case 1:  statement;
           break;
  case 10: statement;
           statement;
           break;
  case 100: statement;
           break;
  default:  printf("Error\n");
           break;
  }
```

```
/* Equivalent if logic */

if (num == 1)
   statement;
else if (num == 10) {
   statement;
   statement;
   }
else if (num == 100)
   statement;
else
   printf("Error\n");
```

VOLUME 3 SUMMARY

Arrays

declaration

```
char    line[256];
float   length[20];
int     table[50][60];
```

reference

```
line[0]
length[num + 1]
table[5][10]
```

index 0 to (size - 1)

strings terminated with '\0'

Increment and decrement operators

adds one	*++x*	*prefix, immediate*
	x++	*postfix, after used*
subtracts one	*--x*	*prefix, immediate*
	x--	*postfix, after used*

do *statement*
while *(expression);*

for (*expr1; expr2; expr3*)
statement

VOLUME 3 LAB

Try as many lab problems as you can.

1. Use a **for** loop to store the numbers 1-25 in an array. Then print the array elements.
2. Write a program to read lines of *standard-input* and write each line to the *standard-output* with the characters in reverse order.
3. Write a program to write the visible characters of the ASCII sequence on the *standard-output*. Have the output characters appear in a rectangular format; eight columns per row.
4. Write a program to read text from a file (via the *standard input*) and output a frequency count for words of different length.

VOLUME 3 LAB - SOLUTIONS

```
/* 3.2.c */
/* Prints the characters in input lines backwards. */

#include <stdio.h>

main()
{
        char line[100];
        int c, i;

        while (1)
          {
          i = 0;
          while (( c=getchar()) != '\n' && c != EOF && i < 100)
              line[i++] = c;

           for (--i; i >= 0; i--)
              putchar( line[i] );
           putchar('\n');

           if ( c == EOF )
             break;
           }
}
```

```
/* 3.4.c */
/* This program reads text from a file and outputs a
 * frequency count for words of different length. */

#include <stdio.h>

main()
{
int c;
int wrdlen [31], numlet=0, i ;

while ( (c=getchar()) != EOF ) {
   if ( c == ' ' || c == '\t' || c == '\n' )
        {
        if  (numlet == 0)
           continue;  /* Disregard sequential white space
                         characters */
        wrdlen [numlet]++ ;
        numlet = 0 ;
        continue ;
        }
   if ( ( c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') ||
              c == '-' )
        numlet++ ;
   }

for ( i=1 ; i<=15 ; i++ )
   printf("%d\t%d \t\t%d\t%d\n",
         i, wrdlen [i], i+15, wrdlen[i+15]);
}
/*  TO EXECUTE WITH filex AS INPUT SOURCE
$ cc 3.4.c
$ a.out <filex

   OR

$ cc 3.4.c -octchar
$ ctchar <filex
*/
```

VOLUME 4

Formatted I/O

printf() - ADDITIONAL FORMATTING

Some other printf() features not formally discussed in this course are:

plus or blank	By default, negative values are preceded by a '–'. To have positive values preceded by a '+', use %+d, %+f, etc. To have positive values preceded by a blank, place a blank after the '%', e.g., % d, % f, etc.
alternate form	To print octal values with a leading 0, use %#o; to print hex values with a leading 0x, use %#x. %#f and %#e will cause a decimal point to be printed even if there is no fractional part.

☞ See manual for more details ☜

VOLUME 4 EXERCISE
printf()

Do these exercises on paper (not on a terminal). Given:

```
#include    <stdio.h>

main()
{
    int    val;
    float  sum;
    char   name[36];
    .
    .
```

1. Write printf() statements that print

    a. the value in val: _____

    b. the string in name: _____

    c. the value of sum / val: _____

2. Write a printf() statement that prints one line with

    - val in a left-justified column at least 8 characters wide,
    - sum in a right-justified column at least 10 characters wide with 3 decimal places.

_____

VOLUME SUMMARY
scanf()

scanf()	
%conversion-char	
c	any 1 char
d, u, o, x, X	integer:decimal,unsigned,octal,hex Precede with l if long,h if short
e, f, g	float: precede with l if double
s	string of non-whites NULL added to destination
All arguments must be addresses. scanf() returns number of fields matched or EOF. Check return code. If scanf() fails: Clear to end of line, field, or record or exit program.	

☞ See manual or Volume 12 Appendix for further information ☜

VOLUME 4 EXERCISE
scanf()

Do these exercises on paper, not on a terminal. For problems 1 and 2, use scanf() statements. Store the return value in the integer 'x'.

```
1    #include   <stdio.h>
2
3    main()
4    {
5         int    x,sum;
6         double  diameter;
7         char    name[81];
8.
9.
```

1. Read a decimal integer into sum.
2. Read a floating point number into diameter.
3. The %s conversion character is used to read a string of characters into an array. Any leading white space on the input stream is skipped over. Then any number of non-white characters are read into the array. A NULL is placed at the end of the string. Assume the input stream contains "5620-Terminal 32100 Layers". If either of the following scanf() statements is used,

   ```
   scanf("%s",&name[0]);
   scanf("%s",name);
   ```

   what will the 'name' array contain?

   Later in this course, the library functions (gets() and fgets()), which read an entire line into an array, are discussed.

   Remember that there is a more detailed treatment of scanf in the Appendix - Volume 12.

VOLUME 4 LAB

1. Write a program that prompts the user for two numbers. Read the numbers using scanf() and print their sum.

2. Write a program that prompts the user to enter in the average daily temperatures for a 7-day period. Store these temperatures in an array. Print the stored temperatures. Then calculate and print the average temperature for the week.

3. As part of the C Language Programming Course Lab Exercises, you will be asked to write programs as part of a Building Block Lab. Try to write these programs before going on to another Course Unit because subsequent Building Block Labs will incorporate your previously written programs into new problem solutions.

   You have recently been retained as an information systems consultant to Hopewell Federal Savings and Loan. Your first contract is to design a menu program that will allow S&L employees to select from various features offered by a proposed S&L Automation System that you will provide.

   a. Write a program named *menu.c* which will prompt the user with the following choices:

      Calculate a monthly mortgage payment.

      Generate a mortgage payment schedule.

      Examine a loan account.

      Examine a savings account.

      Look up a telephone number.

   The user should make a selection from the menu and the program should echo back the selection made. Error checking should be provided for invalid selections.

VOLUME 4 LAB - SOLUTIONS

```
/* 2.2.c */
/* Prompts user for two numbers and prints their sum */

#include        <stdio.h>

main()
{
        int   x;
        float    num1, num2;

        printf("This program will print the sum of 2 numbers.\n");
        printf("First number: ");
        while (scanf("%f",&num1) != 1) {
            while (getchar() != '\n')  /* Clear line */
                ;
            printf("Illegal input.  Try again: ");
            }
        printf("Second number: ");
        while (scanf("%f",&num2) != 1) {
            while (getchar() != '\n')  /* Clear line */
                ;
            printf("Illegal input.  Try again: ");
           }
     printf("\n%.2f + %.2f = %.2f\n",num1, num2, num1 + num2);
}
```

VOLUME 4 LAB - SOLUTIONS

```
/* 4.3a.c */
/* Print a menu & read user's selection */

#include <stdio.h>

main()
{
        int selection=0, c;

        while (selection != 6) {
          printf("\n\n\t\t\tWHAT WOULD YOU LIKE TO DO NOW ?");

          printf("\n\n\t\t1 - Calculate a monthly mortgage payment.");
          printf("\n\n\t\t2 - Generate a mortgage payment schedule.");
          printf("\n\n\t\t3 - Examine a loan account.");
          printf("\n\n\t\t4 - Examine a savings account.");
          printf("\n\n\t\t5 - Look up a telephone number.");
          printf("\n\n\t\t6 - EXIT");
          printf("\n\n\tPlease make your selection (1-6): ");

          while ( scanf("%d", &selection) != 1 )
                while (( c = getchar()) != '\n')
                        ;                       /* flush stdin */
          printf ("\n\n\tYour choice was:\t");

          switch ( selection ) {
                case 1:  printf("Monthly payment calculation\n\n");
                         break;
                case 2:  printf("Mortgage payment schedule\n\n");
                         break;
                case 3:  printf("Examine loan account\n\n");
                         break;
                case 4:  printf("Examine savings account\n\n");
                         break;
                case 5:  printf("Telephone lookup\n\n");
                         break;
                case 6:  printf ("EXIT\n");
                         break;
                default: printf("BAD SELECTION\n\n");
                         break;
                }
        }
}
```

VOLUME 5

Operators and Precedence

OVERVIEW OF BITWISE OPERATORS

&	bitwise AND
|	bitwise OR
^	bitwise EXCLUSIVE OR
~	one's complement
<<	left shift
>>	right shift

ONE'S COMPLEMENT OPERATOR~

~integer-expression

- The ~ is a unary operator
- 0's and 1's are reversed
- Example:

```
num = 5;

/* Assume 32-bit word */

 num    00000000000000000000000000000101

~num    11111111111111111111111111111010
```

BIT OPERATIONS WITH MASK

- An integer may be used to hold many true/false values.
- Individual bits in the integer are assigned a meaning.
- Mask: pattern of bits used to test and change the integer.
- Masks are often #defined.

```
/* Masks used in a hypothetical i/o package */
#define READ   1        /* 0001 */
#define WRITE  2        /* 0010 */
#define READ_WRITE  3       /* 0011 */
#define EOF    4        /* 0100 */
#define ERROR 8         /* 1000 */

short int       status; /* Bits indicate how file was opened, */
                /* if eof was reached or an error occurred */

/* Turn bit(s) on: */
                status = status | WRITE;    /* status |= WRITE */

/* Test bit(s): */
                if ((status & READ_WRITE) == READ_WRITE)
                        statement

/* Turn bit(s) off: */
                status = status & ~ERROR;
```

VOLUME 5 SUMMARY

Bitwise Operators

expression **op** expression

&	bitwise AND
|	bitwise OR
^	bitwise EXCLUSIVE OR
<<	left shift
>>	right shift

Unary Operators

-expression	arithmetic negation
!expression	logical negation
~expression	one's complement
sizeof(*object*)	size
(type)expression	*cast*

Comma Operator

FORMAT:

expression , expression

ACTION:

Both expressions are evaluated. The value of the COMPOUND expression is the value of the right-hand expression.

VOLUME 5 LAB

1. Write a program that will take an integer and change it as follows:

   zero out(turn off) bits 11, 9, 7, 2 and 0

   turn on bits 8, 6, 5 and 3

   Leave the other bits alone. A test value of octal 027207 may be used. Doing the above will result in an octal value of 22552.

2. Modify the previous program to accept the integer interactively. (Remember that it must be an octal constant.) Stop input upon input of a special character (programmer choice).

3. Enter an octal number from the terminal and print it out, shifted one bit position right and one position left.

VOLUME 5 LAB - SOLUTIONS

```
/*
    lab - 5.3
    enter octal number from terminal
    and shift left and shift right
*/

#include <stdio.h>

main()
{
     int x ,  ret;

     while(1)
       {
       printf("Please enter an octal number (0 for escape)\n");
       ret = scanf("%o",&x);
       if( ret == 0 )
           {
           printf("Format Mismatch\n");
           break;
           }
        if( x == 0 ) break;
        printf("%o shifted 1 postion left equals %o\n", x , x<<1);
        printf("%o shifted 1 postion right equals %o\n", x , x>>1);
        }
}
```

VOLUME 6

Functions and Storage Classes

VOLUME 6 EXERCISE
FUNCTION CALLS and ARGUMENTS

3. What will this program print?

```
main()
{
        int     x = 5;
        square(x);
        printf("main: x is %d\n",x);
}

square(val)
int   val;
{
        val = val * val;
        printf("%d\n",val);
}
```

THE ROLE OF THE STACK

- Memory used when a function is called
- Grows and shrinks as functions are called or exited

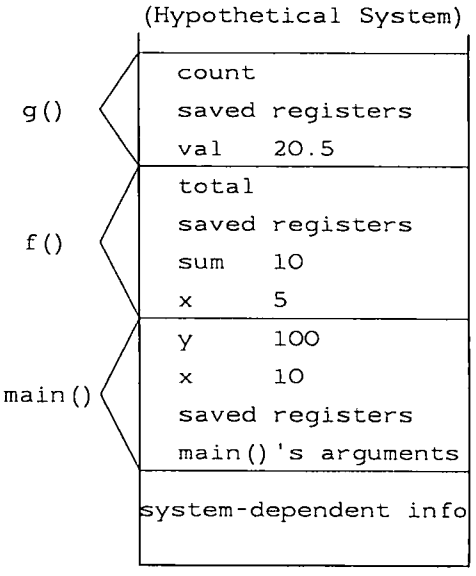
```
main()
{
        int     x=10, y=100;

        f(5,10);
        .
}

f(x, sum)
int     x, sum;
{
        double total;

        g(20.5);
        .
}

g(val)
double val;
{
        int     count;
        .
}
```



VOLUME 6 EXERCISE
FUNCTION ARGUMENTS and RETURN VALUES

4. Write argument declaration statements next to the pointing hands (☞) that correspond with the function calls in main().

```
main()
{
        int     line[100];
        adjust(line[0]);
        sum(line);   /* or sum(&line[0]); */
}

adjust(arg)
☞
{
    ...
}

sum(arg)
☞
{
    ...
}
```

STORAGE CLASS SUMMARY

Class	Scope	Life	Storage	Initialize Arrays, Structs	Default Value
automatic	block	block active	stack	no	undefined
register[1]	block	block active	machine register	no	undefined[2]
external[3]	declaration to end-of-file	permanent	data area	yes	0
static external[4]	declaration to end-of-file	permanent	data area	yes	0
static internal	block	permanent	data area	yes	0

1. Speed advantage
2. For function arguments, value passed
3. Can be accessed from other files
4. Cannot be accessed from other files

VOLUME 6 SUMMARY

Passing array address to a function:

```
main()
{
   char  line[80];
   f(line); /* f(&line[0]); */
   .
}

f(str)
char str[];
{
   .
}
```

VOLUME 6 LAB

Do problem 1 first. If you finish, choose one or more of the other problems to work on. Note that problems 4(a) and 5 are exercises that do not require a terminal. Be sure to complete Problem 6, part of our Building Block Lab.

1. The six problems listed below are similar. Each requires a program to be written with two functions, main() and max(). Two variables should be initialized in main() and passed to max() which finds the larger of the two.

   a. Let max() accept two integers and print the larger one.

   b. Let max() accept two integers and return the value of the larger one to main(). Let main() print the larger one.

   c. Let max() accept two doubles and return a double. Let main() print the value returned.

   d. Let max() accept two doubles and store the larger value in an external variable. Let main() print its value.

   e. Same as (c), but put main() and max() in separate files. See the summary page on the left for directions on compiling these files.

   f. Same as (d), but put main() and max() in separate files.

2. Passing an Array Address to a Function

   Write a function stringlen() which returns the length of a string (not counting the NULL) whose address is passed to it. Here is one way to test the function:

```
main()
{
    static char info[] = "Test String";
    int    len;
    len = stringlen(info);
    printf("Length of info is %d\n",len);
}
```

   The library function strlen(), available with most C compilers, accomplishes this same task. It is discussed later in the course.

Convince yourself that it works. Remember that there are separate instances of the variables *n* and *i* for each call. Also remember that % is the remainder operator.

```
/* The %d part of printf() */

void printd(n)
int  n;
{
 int  i;
 if (n < 0) {
     putchar('-');
     n = -n;
     }
 if ( (i = n/10) != 0)
     printd(i);
 putchar(n % 10 + '0');
}
```

b. Write a recursive function that returns the factorial of a positive integer.
5 factorial == 5 * 4 * 3 * 2 * 1.

c. Write a recursive function to print a number in the Fibonacci series. The Fibonacci series works as follows:

fib(1) == 1
fib(2) == 1

for n > 2, fib(n) == fib(n - 1) + fib(n - 2)
The series looks like: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

VOLUME 6 LAB

6. Building Block Lab Problem

In Problem 3a of Volume 4, you wrote a menu program to print a menu, prompt a user for a selection, read the user's selection and echo back the user's choice.

Write a new version of that program in which the menu and prompt printing are done by one function while another function reads and returns the user's choice.

VOLUME 6 LAB - SOLUTIONS

```
/* 6.1b.c */
/* main() initializes two integers and passes them to max(). */
/* max() returns the larger one and main() prints it. */

#include      <stdio.h>

main()
{
        int     one = 57, two = 208, larger;
        larger = max(one,two);   /* printf("%d\n",max(one,two); */
        printf("%d\n",larger);
}

/* Returns the larger of two arguments */
max(a,b)
int    a,b;
{
        if (a > b)     /* return ( a > b ? a : b ) */
              return (a);
        else
              return (b);
}
```

```
/* 6.1d.c */
/* main() initializes two doubles and passes them to max(). */
/* max() returns the larger one and main() prints it. */

#include        <stdio.h>
double   larger;

main()
{
        double  one = 57.0, two = 208.0, max() ;

        max(one,two);
        printf("%f\n",larger);
}

/* Assigns the larger of two arguments to variable 'larger' */
double   max(a,b)
double   a,b;
{
        if (a > b)          /* larger = ( a > b ? a : b ) */
              larger = a;
        else
              larger = b;
}
```

VOLUME 6 LAB - SOLUTIONS

```
/* 6.1f1.c */
/* main() initializes two doubles and passes them to max(). */
/* max() returns the larger one and main() prints it. */

#include      <stdio.h>
double  larger;

main()
{
        double    one = 57.0, two = 208.0, max() ;

        max(one,two);
        printf("%f\n",larger);
}
```

```
/* 6.1f2.c */
/* Assigns the larger of two arguments to variable 'larger' */
extern  double      larger;

double  max(a,b)
double  a,b;
{
        if (a > b)          /* larger = ( a > b ? a : b ) */
              larger = a;
        else
              larger = b;
}
```

```
/* 6.3.c - Prints address of various automatics, */
/* externals, statics and function parameters */

#include <stdio.h>
int     ext_int;
double  ext_double;

main()
{
        void        f();
        int         m_auto1;
        char        m_auto2;

        /* Use %o or %x to see addresses in octal or hex */
        printf("main: Address of ext_int is %d\n",&ext_int);
        printf("main: Address of ext_double is %d\n",&ext_double);
        printf("main: Address of m_auto1 is %d\n",&m_auto1);
        printf("main: Address of m_auto2 is %d\n",&m_auto2);
        f(m_auto1);
}

void f(f_param)
int     f_param;
{
        void        g();
        int         f_auto1, f_auto2;
        printf("f: Address of f_param is %d\n",&f_param);
        printf("f: Address of f_auto1 is %d\n",&f_auto1);
        printf("f: Address of f_auto2 is %d\n",&f_auto2);
        g(f_auto1);
}
```

```
/* 6.4b.c */
/* Prints the factorial of a number supplied by the user */

#include <stdio.h>

main()
{
        int         num;
        double      fact();

        printf("This program will print the factorial of a number.\n");
        num = get_pos_int();
        printf("%d factorial is %g\n", num, fact((double)num));
        /* %g will use scientific notation for large numbers */
}

/* Recursive function which returns factorial of the */
/* argument passed. doubles are used to store larger
/* numbers than an int could. */
double  fact(x)
double  x;
{
        /* Could check here to make sure number does not */
        /* exceed largest possible value that a double can */
        /* hold the factorial of. */
        if (x == 1)
                return(1);
        else
                return(x * fact(x - 1));
}
```

VOLUME 6 LAB - SOLUTIONS

```
/* 6.4c.c */
/* Prints the n-th number in the Fibonnaci Series. n is supplied */
/* by the user. */

#include <stdio.h>

main()
{
 int      num;
 double      fact();

 printf("This program will print a number ");
 printf("in the Fibonnaci Series.\n");
 printf("Which number in the series would you like to see?\n\n");
 num = get_pos_int();
 printf("Number %d in the Fibonnaci Series is %d\n", num, fib(num));
}

/* Recursive function which returns n-th number in Fibonnaci Series */
fib(n)
int   n;
{
    if (n == 1 || n == 2)
          return(1);
    else
          return(fib(n - 1) + fib(n - 2));
}
```

VOLUME 6 LAB - SOLUTIONS

```
/* 6.5 */


i = 1, j = 1
NEXT i = 11
LAST i = 22
NEW i = 7

i = 1, j = 2
NEXT i = 23
LAST i = 46
NEW i = 10

i = 1, j = 3
NEXT i = 47
LAST i = 94
NEW i = 14
```

VOLUME 6 LAB - SOLUTIONS

```
/* Print S&L System features menu with user prompt */
void prtmenu()
{
        printf("\n\n\t\t\tWHAT WOULD YOU LIKE TO DO NOW ?");
        printf("\n\n\t\t1 - Calculate a monthly mortgage payment.");
        printf("\n\n\t\t2 - Generate a mortgage payment schedule.");
        printf("\n\n\t\t3 - Examine a loan account.");
        printf("\n\n\t\t4 - Examine a savings account.");
        printf("\n\n\t\t5 - Look up a telephone number.");
        printf("\n\n\t\t6 - EXIT");
        printf("\n\n\tPlease make your selection (1-6): ");
}

/* Return an integer read from the user terminal */
/* Flush bad data */
int getselect()
{
        int c, i;
        if ( scanf("%d", &i) != 1 )
          while (( c = getchar()) != '\n')
                ;               /* flush stdin */
        return (i);
}
```

VOLUME 7

The C Preprocessor and Program Organization

ORGANIZATION OF PROGRAM FILES

The next page illustrates the typical way of organizing the files that make up a program. The header file projX.h contains #define statements, declarations, and a typedef statement used by the .c files. Note the defs.c file — it is a good practice to put external variables accessed by several files in their own source file. Functions should not appear in the file. The file containing main() is often called main.c for easy identification. Notice the calc.c and bufctl.c files. The functions within each of these files are related and represent subsystems.

External variable definitions should be in a separate source file (like defs.c).

Because it is common practice, function declarations in header files shown in this text will be preceded by the keyword extern even though it is not necessary.

VOLUME 7 EXERCISE
PREPROCESSOR

1. The C compiler finds an error on line 5. Why?

```
1  #define  LINELEN     80;
2
3  main()
4  {
5         char   line[LINELEN];
6         int    x;
7
```

2. What is wrong with the ISDIGIT macro? Fix the macro.

```
1  #define  ISDIGIT(C)   return((C) >= '0' && (C) <= '9')
2
3  main()
4  {
5         int    input, digits = 0;
6
7         input = getchar();
8         if (ISDIGIT(input))
9               digits ++;
10
```

VOLUME 7 EXERCISE - ANSWERS
PREPROCESSOR

1. Line 1 should not end in a semicolon.

2. Take out return.

WHAT IS A LIBRARY?

- Collection of shared functions
- Supplied with system or created by programmer
- Library functions usually supplied with C compilers:

  Input/output - "Standard I/O"

  String handling

  Character handling

  Memory allocation

  General utilities

  Math functions

For example, use:

```
$ cc prog.c -lm          to use math functions
$ cc prog.c -lcurses     to use curses functions
```

EXPONENTIAL, LOG, POWER AND SQUARE ROOT FUNCTIONS

SYNOPSIS

```
#include   <math.h>

double exp(x)                /* e^x */
double x;

double log(x)                /* natural log of x */
double x;

double log10(x)              /* log base 10 of x */
double x;

double pow(x,y)              /* x^y */
double x,y;

double sqrt(x)               /* square root of x */
double x;
```

EXAMPLES

```
1    #include      <stdio.h>
2    #include      <math.h>
3    main()
4    {
5            double   z = 77.9;
6            printf("%g  %g  ",exp(z), log(z));
7            printf("%g  %g  ",log10(z), pow(z,5.0));
8            printf("%g\n",sqrt(z));
9    }
$ cc prog.c -lm
$ a.out
6.78485e+33  4.35543  1.89154  2.86871e+09  8.8261
```

VOLUME 7 SUMMARY

Compilation Steps

C preprocessor → compiler → assembler → linker

UNIX cc(1) option	produces	contains
-P	prog.i	preprocessor output
-S	prog.s	assembly code
-c	prog.o	object code machine code,not linked
-O	a.out	optimized code

☞ See manual for other C preprocessor features, cpp(1) on UNIX Systems ☜

Preprocessor

Symbolic Constants

```
#define    identifier      token-string
#define    LINELEN         80
```

Macros

```
#define    identifier(arg[,arg]...)
                           token-string
#define    CUBE(X)         ((X) * (X) * (X))
```

Keep macros short and simple. Parenthesis if used with operators.

To debug on UNIX: $ cc -P prog.c; cat prog.i

Header Files, Include Files

```
#include  <file.h>        /* in standard place */
#include  "file.h"        /* local header file */
```

Conditional Compilation

```
#if constant-expression   /* or #ifdef CONSTANT */
                          /* or #ifndef CONSTANT */
C and/or preprocessor statements
#else
C and/or preprocessor statements
#endif
```

VOLUME 7 LAB

The problems in this lab are divided into three categories: the Preprocessor, Libraries, and more work on your Building Block Lab. Problem No. 2 in the Preprocessor Section does not require a terminal. Try as many problems as you can.
Be sure to write the Building Block Lab program.

Preprocessor

1. Write a program which generates a table of miles and the corresponding kilometers:

   a. Let the miles be 0, 10, 20, 30, ... up to a limit defined by a constant UPPER. Define UPPER to be 60. Write a macro KM to calculate the equivalent kilometers.

      1 mile = 1.6129 kilometers

   b. Move the #define statements for UPPER and KM into a separate file called metric.h. Change UPPER to 120. Adjust your source file so that metric.h is included when it is compiled. Print the new table. After printing the table, print a statement which reports the upper speed limit in the eastern United States in kilometers (55 mph).

   c. Look at the output of the preprocessor. On UNIX Systems:

      ```
      $ cc -P prog.c
      $ cat prog.i
      ```

      The many lines at the beginning of the output are from the stdio.h file.

VOLUME 7 LAB

```
 1  /*    @(#)stdio.h      2.7      */
 2  #ifndef _NFILE
 3  #define _NFILE      20
 4
 5  #if u370
 6  #define BUFSIZ      4096
 7  #endif
 8  #if vax || u3b || u3b5
 9  #define BUFSIZ      1024
10  #endif
11  #if pdp11
12  #define BUFSIZ      512
13  #endif
14
15  /* buffer size for multi-character */
16  /* output to unbuffered files */
17  #define _SBFSIZ 8
18
19  typedef struct {
20  #if vax || u3b || u3b5
21  int         _cnt;
22  unsigned char      *_ptr;
23  #else
24  unsigned char      *_ptr;
25  int         _cnt;
26  #endif
27  unsigned char      *_base;
28  char        _flag;
29  char        _file;
30  } FILE;
31
32  /*
33  * _IOLBF means that a file's output
34  * will be buffered line by line
35  * In addition to being flags, _IONBF,
36  * _IOLBF and _IOFBF are possible
37  * values for "type" in setvbuf.
38  */
39  #define _IOFBF              0000
40  #define _IOREAD             0001
41  #define _IOWRT              0002
42  #define _IONBF              0004
43  #define _IOMYBUF            0010
44  #define _IOEOF              0020
45  #define _IOERR              0040
```

VOLUME 7 LAB

3. These programs make use of the math libraries.

   a. Assign a value to a variable of type double. Then print its square root. Remember to link in the math library on the command line.

   b. Same as the previous problem, but get a value for the variable interactively.

4. Building Block Lab (be sure to write these programs)

   a. Your next assignment is to provide a program that will calculate the monthly payment for a fixed rate, constant payment mortgage.

      The periodic payment on a constant payment mortgage can be calculated by the formula:

$$Payment = \frac{P \cdot R}{1 - \frac{1}{(1 + R)^N}}$$

      where:

      P = Principal

      R = Interest rate per payment period as a decimal (Nominal rate ÷ 12 for monthly payments)

      N = Number of payments over life of mortgage

      Write a program to calculate the payment given the principal, interest rate, and years as interactive input.

VOLUME 7 LAB - SOLUTIONS

```
/* 7.1a.c */
/* Prints table of miles and kilometers */

#include  <stdio.h>
#define   UPPER       60
#define   KM(MILES)   (1.6129 * (MILES))

main()
{
   int   miles;

   for (miles = 0; miles <= UPPER; miles += 10)
       printf("%8d mi %8.2f km\n",miles,KM(miles));
}
```

VOLUME 7 LAB - SOLUTIONS

```
/* 7.2 */
                    1 /*   @(#)stdio.h    2.7     */
          cond. comp.  2                    #ifndef _NFILE
             constant  3                    #define _NFILE              20
                       4
          cond. comp.  5                    #if u370
             constant  6                    #define BUFSIZ              4096
                       7                    #endif
          cond. comp.  8                    #if vax || u3b || u3b5
             constant  9                    #define BUFSIZ              1024
                      10                    #endif
          cond. comp.  11                   #if pdp11
             constant  12                   #define BUFSIZ              512
                      13                    #endif
                      14
                      15                    /* buffer size for multi-character */
                      16                    /* output to unbuffered files */
             constant  17                   #define _SBFSIZ 8
                      18
               typedef  19                  typedef struct {
          cond. comp.  20                   #if vax || u3b || u3b5
                      21                    int                 _cnt;
                      22                    unsigned char       *_ptr;
                      23                    #else
                      24                    unsigned char       *_ptr;
                      25                    int                 _cnt;
                      26                    #endif
                      27                    unsigned char       *_base;
                      28                    char                _flag;
                      29                    char                _file;
                      30                    } FILE;
                      31
                      32                    /*
                      33                    * _IOLBF means that a file's output
                      34                    * will be buffered line by line
                      35                    * In addition to being flags, _IONBF,
                      36                    * _IOLBF and _IOFBF are possible
                      37                    * values for "type" in setvbuf.
                      38                    */
             constant  39                   #define _IOFBF              0000
             constant  40                   #define _IOREAD             0001
             constant  41                   #define _IOWRT              0002
             constant  42                   #define _IONBF              0004
             constant  43                   #define _IOMYBUF            0010
             constant  44                   #define _IOEOF              0020
             constant  45                   #define _IOERR              0040
```

VOLUME 7 LAB - SOLUTIONS

```
/* 7.3a.c */
/* Finds the square root of a variable */

#include  <math.h>
#include  <stdio.h>

main()
{
      double    num;

      num = 4567.99;
      printf("The square root of %.2f is %.2f\n",sqrt(num));
}
```

VOLUME 7 LAB - SOLUTIONS

```
/* 7.4a.c - Calculate monthly payment on mortgage */

#include <stdio.h>
#include <math.h>

main()
{
     double bal;          /* principal */
     double pmt;          /* monthly payment */
     double intyr;        /* annual interest rate */
     double intmo;        /* monthly interest rate */
     short npmts;         /* number of payments */
     short nyears;        /* number of years */

     printf("Enter principal (e.g., 82500.00): ");
     scanf("%lf", &bal);

     printf("Enter annual interest rate (e.g., 16.25): ");
     scanf("%lf", &intyr);

     printf("Enter number of years: ");
     scanf("%hd", &nyears);

     printf("\n\nPrincipal=$%.2f\tInterest=%.4f%%\tYears
             =%d\n\n", bal, intyr, nyears);

     intyr /= 100.;
     intmo  = intyr / 12.;
     npmts  = nyears * 12;

     pmt = bal * (intmo / (1. - pow(1. + intmo, -(double)npmts)));

     printf("Monthly payment = $%.2f\n", pmt);
}
```

```
/* 7.4b.c (cont.) */

intyr /= 100.;
intmo  = intyr / 12.;
npmts  = nyears * 12;

pmt = bal * (intmo / (1. - pow(1. + intmo, -(double)npmts)));

printf("%8s %10s %10s %10s %10s\n",          /* Print Headings */
        "payment", "total", "interest", "principal", "balance");
printf("%8s %10s %10s %10s\n",
        "number", "payment", "payment", "payment");
printf("%8s %10s %10s %10s %10.2f\n",
        "", "", "", "", bal);

for (i = 1; i <= npmts; ++i) {              /* Print scheduled */
        intpmt = bal * intmo;               /* payment data */
        if (i < npmts)
           prinpmt = pmt - intpmt;
        else
           prinpmt = bal;
        bal -= prinpmt;
        pmttot += intpmt+prinpmt, intpmttot += intpmt;
        prinpmttot += prinpmt;
        printf("%8d %10.2f %10.2f %10.2f %10.2f\n",
                i, intpmt + prinpmt, intpmt, prinpmt, bal);
        }
printf("%8s %10s %10s %10s\n",                  /* Print Totals */
        "","========","========","========");
        printf("%8s %10.2f %10.2f %10.2f\n",
                "", pmttot, intpmttot, prinpmttot);
}
```

```
/* menu.c  (cont.) */
/* Print S&L System features menu with user prompt */
void prtmenu()
{
        printf("\n\n\t\t\tWHAT WOULD YOU LIKE TO DO NOW ?");
        printf("\n\n\t\t1 - Calculate a monthly mortgage payment.");
        printf("\n\n\t\t2 - Generate a mortgage payment schedule.");
        printf("\n\n\t\t3 - Examine a loan account.");
        printf("\n\n\t\t4 - Examine a savings account.");
        printf("\n\n\t\t5 - Look up a telephone number.");
        printf("\n\n\t\t6 - EXIT");
        printf("\n\n\tPlease make your selection (1-6): ");
}

/* Return an integer read from the user terminal */
/* Flush bad data */
int getselect()
{
        int c, i;
        if ( scanf("%d", &i) != 1 )
        while (( c = getchar()) != '\n')
            ;       /* flush stdin */
        return (i);
}
```

VOLUME 7 LAB - SOLUTIONS

```
/* 7.4c */
/* pmttbl - A function to  prompt user for mortgage loan data */
/*          and print table of payments on mortgage */

#include <stdio.h>
#include <math.h>

pmttbl()
        {
        double intmo;       /* monthly interest */
        double intyr;       /* annual interest */
        double bal;         /* balance remaining */
        double pmt;         /* monthly payment */
        double prinpmt;     /* payment allocated to principal */
        double intpmt;      /* payment allocated to interest */
        short i;            /* loop index */
        short npmts;        /* number of payments */
        short nyears;       /* number of years */
        double pmttot=0, intpmttot=0, prinpmttot=0; /* totals */

        printf("Enter principal (e.g., 82500.00): ");
        scanf("%lf", &bal);

        printf("Enter annual interest rate (e.g., 16.25): ");
        scanf("%lf", &intyr);

        printf("Enter number of years: ");
        scanf("%hd", &nyears);

        printf("\n\nPrincipal=%.2f  Interest=%.4f%%  Years=%d\n\n",
                        bal, intyr, nyears);
```