7.12 Write the following function that searches the \( n \) bytes beginning with \( s \) for the character \( c \), where \( n \) is the number of bytes that \( s \) has to be incremented before it points to the null character ‘\0’. If the character is found, a pointer to it is returned; otherwise return NULL:

\[
\text{char* chr(char* s, char c)}
\]

7.13 Write the following function that returns the sum of the floats pointed to by the first \( n \) pointers in the array \( p \):

\[
\text{float sum(float* p[], int n)}
\]

7.14 Write the following function that changes the sign of each of the negative floats pointed to by the first \( n \) pointers in the array \( p \):

\[
\text{void abs(float* p[], int n)}
\]

7.15 Write the following function that indirectly sorts the floats pointed to by the first \( n \) pointers in the array \( p \) by rearranging the pointers:

\[
\text{void sort(float* p[], int n)}
\]

7.16 Implement the Indirect Selection Sort using an array of pointers. (See Problem 6.19 on page 144 and Example 7.17 on page 170.)

7.17 Implement the Indirect Insertion Sort. (See Problem 6.18 on page 144 and Example 7.17 on page 170.)

7.18 Implement the Indirect Perfect Shuffle. (See Problem 6.29 on page 145.)

7.19 Rewrite the \texttt{sum()} function (Example 7.18 on page 171) so that it applies to functions with return type \texttt{double} instead of \texttt{int}. Then test it on the \texttt{sqrt()} function (defined in \texttt{<math.h>}) and the reciprocal function.

7.20 Apply the \texttt{riemann()} function (Problem 7.4 on page 173) to the following functions defined in \texttt{<math.h>}: 
\[
a. \texttt{sqrt()}, on the interval \([1, 4])
\]
\[
b. \texttt{cos()}, on the interval \([0, \pi/2])
\]
\[
c. \texttt{exp()}, on the interval \([0, 1])
\]
\[
d. \texttt{log()}, on the interval \([1, e])
\]

7.21 Apply the \texttt{derivative()} function (Problem 7.5 on page 175) to the following functions defined in \texttt{<math.h>}: 
\[
a. \texttt{sqrt()}, at the point \( x = 4 \)
\]
\[
b. \texttt{cos()}, at the point \( x = p/6 \)
\]
\[
c. \texttt{exp()}, at the point \( x = 0 \)
\]
\[
d. \texttt{log()}, at the point \( x = 1 \)
\]

7.22 Write the following function that returns the product of the \( n \) values \( f(1), f(2), \ldots, f(n) \). (See Example 7.18 on page 171.)

\[
\text{int product(int (*pf)(int k), int k, int n)}
\]

7.23 Implement the Bisection Method for solving equations. Use the following function:

\[
\text{double root(double (*pf)(double x), double a, double b, int n)}
\]

Here, \( pf \) points to a function \( f \) that defines the equation \( f(x) = 0 \) that is to be solved, \( a \) and \( b \) bracket the unknown root \( x \) (i.e., \( a \leq x \leq b \)), and \( n \) is the number of iterations to use. For example, if \( f(x) = x^2 - 2 \), then \( \text{root}(f, 1, 2, 100) \) would return 1.414213562373095 (= \( \sqrt{2} \)), thereby solving the equation \( x^2 = 2 \). The Bisection Method works by repeatedly bisecting the interval and replacing it with the half that contains the root. It checks the sign of the product \( f(a) f(b) \) to determine whether the root is in the interval \([a, b]\).

7.24 Implement the Trapezoidal Rule for integrating a function. Use the following function:

\[
\text{double trap(double (*pf)(double x), double a, double b, int n)}
\]

Here, \( pf \) points to the function \( f \) that is to be integrated, \( a \) and \( b \) bracket the interval \([a, b]\) over which \( f \) is to be integrated, and \( n \) is the number of subintervals to use. For example, the