HOMEWORK 3

3-1 Write a program that translates an alphabetic phone number into numeric form:

Enter phone number: CALLATT
2255288

(In case you don’t have a telephone nearby, here are the letters on the keys: 1=.QZ, 2=ABC, 3=DEF, 4=GHI, 5=JKL, 6=MNO, 7=PRS, 8=TUV, 9=WXY.) If the original phone number contains nonalphabetic characters (digits or punctuation, for example), leave them unchanged:

Enter phone number: 1-800-COLLECT
1-800-265-5328

You may assume that any letters entered by the user are upper case.

3-2 Write a program that reads a 5 x 5 array of integers and then prints the row sums and the column sums:

Enter row 1: 8 3 9 0 10
Enter row 2: 3 5 17 1 1
Enter row 3: 2 8 6 23 1
Enter row 4: 15 7 3 2 9
Enter row 5: 16 14 2 6 0

Row totals: 30 27 40 36 28
Column totals: 34 37 37 32 21

3-3 Write a program that reads a message, then prints the reversal of the message. The output of the program should look like this:

Enter a message: Don’t get mad, get even.
Reversal is: .neve teg, dam teg ’noD

Hint: Read the message on character at a time (using getchar) and store the characters in an array. Stop reading when the array is full or the character read is ‘\n’.

Revise the program to use a pointer instead of an integer to keep track of the current position in the array.
3-4 The following structures are designed to store information about objects on a graphics screen. A point structure stores the x and y coordinates of a point on the screen. A rectangle structure stores the coordinates of the upper left and lower right corners of a rectangle.

\begin{verbatim}
struct point { int x, y; }
struct rectangle { struct point upper_left, lower_right; }
\end{verbatim}

Write a program that calls functions performing the following operations on a rectangle structure \( r \) passed as an argument:

(a) Compute the area of \( r \).
(b) Compute the center of \( r \), returning it as a point value.
(c) Move \( r \) by \( x \) units in the \( x \) direction and \( y \) units in the \( y \) direction, returning the modified version of \( r \). (\( x \) and \( y \) are additional arguments to the function.)
(d) Determine whether a point \( p \) lies within \( r \), returning TRUE or FALSE. (\( p \) is an additional argument of type struct point.)

The output of the program should look like this:

Enter a upper left point : 0 0
Enter a lower right point : 50 50
rect{(0, 0), (50, 50)}
-area : 2500
-center point : (25, 25)
Enter a direction : 20 20
Move to a new point : rect{((20, 20), (70, 70)}
Enter a point \( p \) : 100 100
\( p \) is not in the rectangle.

3-5 Write the following function:

\begin{verbatim}
int *find_middle (int a[], int n);
\end{verbatim}

When passed an array \( a \) of length \( n \), the function will return a pointer to the array’s middle element. (If \( n \) is even, choose the middle element with the larger index; for example, if \( n=4 \), the middle element is \( a[2] \), not \( a[1] \).)

Write a program that calls the find_middle function.
The output of the program should look like this:

Enter the array : 2 4 1 3 6 7
The middle element : 3

Note: Write a makefile that generates the executables of all the programs, and include your comments in the source files.