1. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int a = 99;
    int b = 0;
    int c = 74;

    if( a || b )
        printf("first\n");
    else
        printf("second\n");

    if( a && c )
        printf("third\n");
    else
        printf("fourth\n");

    if( !a )
        printf("fifth\n");
    else
        printf("sixth\n");

    if( (a && b) || c )
        printf("seventh\n");
    else
        printf("eighth\n");

    if( !c || !b )
        printf("nineth\n");
    else
        printf("tenth\n");
}
```

2. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int k, count = 0, mysum = 0;

    k = 5;

    while( k <= 60 )
    {
        if( k%10 == 0 )
        {
            mysum += k;
            printf("%d  ", k);
        }

        count += 1;

        if( count == 4 )
        {
            printf("%d\n", mysum);
            mysum = 0;
            count = 0;
        }

        k = k + 5;
    }
}
```

3. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int r, c;

    r = 2;
    while( r < 5 )
    {
        c = 1;
        while( c < 5 )
        {
            printf("%2d, ", r*c );

            if(r == c)
            {
                printf("\n");
                break;
            }
            c += 1;
        }

        r += 1;
    }

}
```

4. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int x, i;

    printf("Enter an integer:  ");  /* assume user enters 50 */
    scanf("%d", &x);

    i = 1;

    while( i <= 5 )
    {
        if( i%2 == 0 )
            x = x + 5;
        else
            x = x - 5;

        printf("%d, ", x);
        i += 1;
    }
}
```

5. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int k, mysum = 0;

    k = 8;

    while( (k > 0) && (mysum < 20) )
    {
        mysum += k;
        printf("%d   %d\n", k, mysum);

        k = k - 1;  /* subtraction */
    }
}
```

6. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int age;

    printf("Enter your age:  ");
    scanf("%d", &age);   /*  assume user enters 15  */

    if( age < 10 )
        printf("child\n");

    if( age < 20 )
        printf("teenager\n");

    if( age < 60 )
        printf("adult\n");

    if( age < 100 )
        printf("old\n");
}
```

7. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int k = 5;

    while( k < 15 )
    {
        if( k%3 == 0 )
        {
            printf("red:  %d\n", k);

            if( k%4 == 0 )
                printf("blue:  %d\n", k);
        }

        k += 1;
    }
}
```

8. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int k = 5, count = 0;

    while( count <= 6 )
    {
        if( (k%5 == 0) && (k%10 != 0) )
            printf("%d   %d\n", count, k);

        count += 1;
        k = k + 5;
    }
}
```

9. Write a program that prompts the user for two integers, a and b, and then prints the integers from a to b, each on a separate line. For those integers that are even, print the word 'even' also on that line. Example:

```
Enter an integer: 4
Enter an integer (>= the first): 9
4 even
5
6 even
7
8 even
9
```

10. Write a program that prompts the user for integers until it receives a positive, odd integer. At this point the program will count down from this number until it reaches zero. Example:

```
Enter an integer: 8
Enter an integer: -3
Enter an integer: 0
Enter an integer: 66
Enter an integer: 7
7
6
5
4
3
2
1
0
```

11. Write a program that prompts the user for two positive integers, a and b, such that $a < b$. The program should count up from a in increments of 1 and down from b in increments of 1 until the two values pass each other. Example:

```
Enter an integer, a:  4
Enter an integer, b (a < b):  10

a = 4, b = 10
a = 5, b = 9
a = 6, b = 8
a = 7, b = 7
```

12. Write a program that prompts the user for two positive integers, which we will call b and e. The program will calculate b^e. That is, the program will raise a positive integer to a positive integer power. Do not use `b**e` or the `pow()` function. Hint: think about how you would explain to a first-grader what b^e means.

13. Write a program that prompts the user for two positive integers and then determines which of them has more divisors. Print which has more divisors or if they have the same number of divisors. You don't need to do any error-checking.

14. Write a program that prompts the user for a positive integer and then, depending on its divisibility by 8 or 12, prints either "first", "second", "third", or "fourth". Use the truth table below to determine what to print.

div by 8	div by 12	Prints
T	T	first
T	F	second
F	T	third
F	F	fourth

15. Write a program that prompts the user for two positive integers, a and b, and then calculates `a * b` without using multiplication. Hint: think about what multiplication is and how we can replace it with addition.

16. Write a program that prompts the user for a positive integer, n, and then prints a sideways pyramid of stars that is n stars high. Example:

```
Enter a positive integer: 5
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

17. Write a program that prompts the user for a positive integer n and then sums the integers in the range of 100 to 200 (inclusive) that are evenly divisible by n. The program will print the final sum. You do not have to do any error-checking.
Example:

```
Enter a positive integer:  13
The sum is 1196
```

18. Write a program that prompts the user for positive integers, only stopping when a negative integer or zero is given. The program should then print out how many of the positive integers were odd.
    Example:

```
Enter a positive integer (0 or negative to stop):  9
Enter a positive integer (0 or negative to stop):  4
Enter a positive integer (0 or negative to stop):  7
Enter a positive integer (0 or negative to stop):  -3
You entered 2 odd integers.
```

19. Write a program that sums the integers in the range of 10 to 100 (inclusive) if the integer is evenly divisible by 4 or 6, but not both. For example, 16 should be part of the sum and 18 should be part of the sum, but not 24. Print the overall sum.
    Hint: Think of the truth table for this.

20. Write a program that prompts the user for a positive integer n and then produces an $n \times n$ multiplication table. You can assume that all values in the table can be printed with no more than three digits You do not have to do any error-checking nor do you need to put row or column labels.
    Example:

```
Enter a positive integer:  4

   1   2   3   4
   2   4   6   8
   3   6   9  12
   4   8  12  16
```

21. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int d[][4] = {{ 3, 8, 9, 4},
                  { 2, 7, 6, 5}};
    int r, c;

    r = 0;
    while( r < 2 )
    {
        c = 2 ;
        while( c < 4 )
        {
            if( d[r][c]%2 == 0 )
                d[r][c-2] = d[r][c-2] + 10;
            else
                d[r][c-2] = 99;

            c += 1;
        }

        r += 1;
    }

    r = 0;
    while( r < 2 )
    {
        c = 0;
        while( c < 4 )
        {
            printf("%3d", d[r][c] );
            c += 1;
        }

        printf("\n");
        r += 1;
    }
}
```

22. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int d[][4] = {{ 13,  7, 9, 1 },
                  {  2,  8, 2, 7 },
                  {  4, 11, 3, 5 }};
    int x, r, c, rows, cols, mod;

    printf("Enter a positive integer:  ");
    scanf("%d", &x);  /* assume user enters 4 */

    rows = sizeof(d) / sizeof(d[0]);
    cols = sizeof(d[0]) / sizeof(int);

    r = 0;
    while( r < rows )
    {
        c = 0;
        while( c < cols )
        {
            mod = d[r][c] % x;
            d[r][c] = mod;
            c += 1;
        }

        r += 1;
    }

    for(r = 0; r < rows; r++)
    {
        for(c = 0; c < cols; c++)
            printf("%3d", d[r][c] );

        printf("\n");
    }
}
```

23. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int d[][3] = {{ 5, 10, 15},
                  {15, 20, 25},
                  {25, 30, 35}};
    int i, k, temp;

    for(i = 0; i < 2; i++)  /* note the range of i */
        for(k = 0; k < 3; k++)
            if( d[i][k] < d[i+1][k] )
            {
                temp = d[i][k];
                d[i][k] = d[i+1][k];
                d[i+1][k] = temp;
            }

    for(i = 0; i < 3; i++)
    {
        for(k = 0; k < 3; k++)
            printf("%2d ", d[i][k]);

        printf("\n");
    }
}
```

24. What does the following program print?

```
#include <stdio.h>

int main(void)
{
                /* 01234567890 */
    char text[] = "hocus pocus";
    int i = 0;

    while( text[i] != ’\0’ )
    {
        if( text[i] == ’c’ )
            printf("b");
        if( text[i] == ’o’ )
            printf("a");
        if( text[i] == ’p’ )
            printf("d");
        if( text[i] == ’s’ )
            printf("a");
        if( text[i] == ’u’ )
            printf("r");
        if( text[i] == ’ ’ )
            printf("ca");

        i++;
    }
}
```

25. What does the following program print?

```
#include <stdio.h>

int fx( int n );

int main(void)
{
    int i = 5;

    while( i < 10 )
    {
        printf("%d:  %d\n", i, fx(i) );
        i += 1;
    }
}

int fx( int n )
{
    int i, mysum = 0;

    i = 1;
    while( i <= n )
    {
        if( n%i == 0 && i%2 == 1 )
            mysum += i;

        i += 1;
    }

    return mysum;
}
```

26. What does the following program print?

```
#include <stdio.h>

void fx(int d[], int cols)
{
    int i;

    for(i = 0; i < cols; i++)
        if( i%2 != 0 )     /* note the comparison */
            d[i] = 0;
}

int main(void)
{
    int a[] = {8, 1, 4, 3, 5};
    int i, sum = 0;

    fx( a, 5 );

    for(i = 0; i < 5; i++)
    {
        sum += a[i];
        printf("%d  %2d\n", a[i], sum);
    }
}
```

27. What does the following program print?

```
#include <stdio.h>

int fx(int a, int b)
{
    return 2*a - b;
}


int main(void)
{
    int i, k;

    for(i = 1; i < 3; i++)
        for(k = 5; k > 2; k--)
            printf("%d, %d, %2d\n", i, k, fx(i, k) );

    if( fx(3, 6) )
        printf("\n first\n");

    if( fx(6, 3) )
        printf("\n second\n");
}
```

28. What does the following program print?

```
#include <stdio.h>

int main(void)
{
    int a[6] = {7, 19, 4, 3, 2, 6};
    int b[6] = {8, 10, 5};
    int i, x, temp;

    printf("Enter an integer:  ");
    scanf("%d", &x);   /* assume user enters 10 */

    for(i = 0; i < 6; i++)
        if( a[i]+b[i] < x )
        {
            temp = b[i];
            b[i] = a[i];
            a[i] = temp;
        }

    for(i = 0; i < 6; i++)
        printf("%2d  %2d\n", a[i], b[i]);
}
```

29. What does the following program print?

```
#include <stdio.h>
#include <string.h>

int main(void)
{
               /*  0123456789  */
    char text[] = "gnoc ioa";
    int i, size;
    size = strlen( text );

    for(i = 0; i < size; i++)
        if(i%2 == 1)
            text[i] = text[i] + 1;

    printf("%s\n\n", text);

    for(i = 0; i < 4; i++)         /* note range of i */
        printf("%d\n", text[i]);   /* uses %d */
}
```

30. What does the following program print?

```
#include <stdio.h>

int main(void)
{
               /* 012345678901234567890123 */
    char text[] = "wtrhoinsg  iasn sewaesry";
    int i = 0;

    while( text[i] != '\0' )
    {
        if( i%2 == 1 )
            printf("%c", text[i]);
        i += 1;
    }
}
```

31. What does the following program print?

```
#include <stdio.h>

int fx(int a, int b);

int main(void)
{
    int i = 3, out;

    while( i < 8 )
    {
        out = fx(i, i+1);
        printf("%d\n", out);
        i += 1;
    }
}

int fx(int a, int b)
{
    int c = 2*a + b;
    return c;
}
```

32. What does the following program print?

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    /*  the ASCII value of a space is smaller than any letter  */
    char word[] = "TZHKKIYSZ ISU FXUXNR";
    int i, length;

    length = strlen( word );

    i = 0;
    while( i < length - 1 )
    {
        if( word[i] < word[i+1] )
            printf("%c", word[i]);

        i += 1;
    }
}
```

33. All of these questions are related to bits and number bases.

(a) Convert the following base-10 number to base-2: 13

(b) Convert the following base-2 number to base-10: 11111

(c) If we have a three-byte number, how many different bit patterns can this number represent? Write it using powers (for example, 5^4).

(d) What are the smallest and largest values in base-10 that can be represented by a 4-bit integer variable if we don't need to represent the sign of a number.

(e) What is the minimum number of bits required to store the base-10 number 22 as an integer variable if we don't need to represent the sign of a number.

34. Finish the program below by adding the code to print the running average, where the average is based on a moving set of three values. For this particular data, the program would print the average of {1, 2, 3}, then the average of {2, 3, 10}, then the average of {3, 10, 9}, and so forth, finishing with the average of {4, 5, 6}. Print the average to three decimal places. The code that you write should not be hard-coded to this particular data except for the fact that there are nine values in the list.

```
#include <stdio.h>

int main(void)
{
    int d[] = {1, 2, 3, 10, 9, 8, 4, 5, 6};
```

35. Write a program that prompts the user 5 times for integer values, printing the current average after each number is entered. Print the average to two decimal places. Example:

```
Enter a number:  5
The average of the first 1 numbers is 5.00
Enter a number:  6
The average of the first 2 numbers is 5.50
Enter a number:  10
The average of the first 3 numbers is 7.00
Enter a number:  3
The average of the first 4 numbers is 6.00
Enter a number:  8
The average of the first 5 numbers is 6.40
```

36. Write a function that receives an integer and returns 1 if this integer is a perfect number and 0 if not. An integer n is a perfect number if the sum of the divisors (excluding n itself) of n are equal to n. Example: 6 is a perfect number since $1 + 2 + 3 = 6$.

37. Write the remainder of the program to find the maximum value for the middle row of the array `data`. Print the max after finding it. Your code should work for any 2D array of `int`s with three rows and four columns, so don't hard-code your program to these specific values.

```
#include <stdio.h>

int main(void)
{
    int data[][4] = {{ 5,  8, 7, 2},
                     { 3, 10, 1, 9},
                     { 6, 12, 5, 0}};
```

38. Write a function that when given two integers, x and y, returns the maximum divisor of x or y, excluding x or y themselves. For example, if the two integers are 8 and 9, the divisors of 8 (excluding 8) are 1, 2, and 4 while the divisors of 9 (excluding 9) are 1 and 3. The maximum is 4.

39. Write a program that stores the following data in an array:

```
9, 7, 4, 2, 11, 16
```

The program should determine and print the sum of the even integers as well as the sum of the odd integers. Your logic should not be hard-coded to these specific values, that is, if I replaced these values with a different set of six integers, your program should still work correctly.

40. Write a program that prompts the user for an integer, `n`, greater than 2. The program should then count how many of the integers in the range of 2 to `n` (inclusive) are NOT prime. You do not have to do any error-checking of the input.

41. Write a program that stores the following string (without the quotation marks):

```
"abc DEF 123 XyZ"
```

The program should count and print out the number of uppercase characters. If I were to replace this string with another, your program should still work.

42. Write a program that stores the string `"47"`. Your program should then convert this to the the integer `47`, without using `atoi()`. You can write your program such that it only works for two-digit integers, but it should work for any two-digit integer. Hint: the conversion can be performed in one line of code.

43. Finish the program below by writing the code to change any number in the array to 99 if, and only if, the values immediately before and after the current number are both even. The first and last elements of the array should not be changed. The array should be processed beginning with the element with index 1. Note that when processing a number, if the value before it has been changed to 99 that is what should be considered, not the value in the initial version of the array. Your program should still work if I changed the array values.

```
#include <stdio.h>
int main(void)
{
    int d[] = {9, 8, 2, 15, 6, 32, 10, 4};
```

44. Write a program that prompts the user for a positive integer, n, and then prints a sideways pyramid of stars that is n stars high. Example:

```
Enter a positive integer: 3
*
* *
* * *
* *
*
```

45. What does the following program print?

```
#include <stdio.h>

void fx(double d[][3], int rows, double* first, double* second, double* third);

int main(void)
{
    double d[][3] = {{1.0, 1.5, 2.0},
                     {1.5, 4.0, 2.5},
                     {2.0, 2.5, 3.0},
                     {2.5, 2.0, 1.0}};
    double f, s, t;

    fx(d, 4, &f, &s, &t);

    printf("final:  %.1f, %.1f, %.1f\n", f, s, t);
}


/* function definition */
void fx(double d[][3], int rows, double* first, double* second, double* third)
{
    int r, c, i = 1;

    *first = d[0][0];
    *second = d[0][0];
    *third = d[0][0];

    printf("%d:  %.1f, %.1f, %.1f\n", i, *first, *second, *third);

    for(r = 0; r < rows; r++)
        for(c = 0; c < 3; c++)
            if( d[r][c] > *first )
            {
                *third = *second;
                *second = *first;
                *first = d[r][c];
                i++;
                printf("%d:  %.1f, %.1f, %.1f\n", i, *first, *second, *third);
            }
}
```

46. What does the following program print?

```
#include <stdio.h>

void fx(int d[], int size);

int main(void)
{
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int i, size;

    size = sizeof( a )/sizeof( int );
    fx( a, size );

    printf("\n");
    for(i = 0; i < size; i++)
        printf("%d  ", a[i]);
}

/* function definition */
void fx(int d[], int size)
{
    int i, k, sum;

    for(i = 0; i < size; i++)
    {
        sum = 0;
        for(k = 0; k <= i; k++)
            sum = sum + d[k];  /* this is the tricky part */

        printf("%d\n", sum);

        if( sum%2 == 0 )
            d[i] = 0;
    }
}
```

47. What does the following program print?

```
#include <stdio.h>

struct address {
    char street[40];
    char state[3];
    int zip;
};

int main(void)
{
    struct address places[4] = {{"123 Main", "TX", 76013},
                                {"456 Oak", "AR", 72201},
                                {"987 Rodeo", "CA", 90212},
                                {"1600 Pennsylvania", "DC", 20500}};
    int i;

    for(i = 0; i < 4-1; i++)
    {
        if( strcmp( places[i].state, places[i+1].state) > 0)
            printf("%s\n", places[i].street);
        else
            printf("%s\n", places[i].state);
    }

    for(i = 0; i < 4-1 ; i++)
    {
        if( (places[i].zip < 75000) && (places[i+1].zip != 72201)  )
            printf("%d\n", places[i].zip);
        else
            printf("no match\n");
    }
}
```

48. What does the following program print?

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char* words[] = {"horse", "rabbit", "fish", "elephant", "squirrel"};
    char text[30];
    int i;

    printf("Enter a word:  ");
    scanf("%s", text);  /*  assume user enters dog  */

    for(i = 0; i < 5; i++)
    {
        printf("%s, %s\n", text, words[i]);

        if( strcmp(text, words[i]) < 0 )
            strcpy(text, words[i]);      /* string copy */
    }

    printf("\n");
    for(i = 0; i < 5; i++)
        printf("%s\n", words[i]);
}
```

49. Write a program that prompts the user for two integers, `lower` and `upper`; we assume that `lower` < `upper`. The program will then read a file, `integers.txt`, which consists of an unknown number of lines, each of which consists of a single integer. The program should produce and print three counts with regards to the integers in `integers.txt`:

    (a) number of integers in `integers.txt` < `lower`

    (b) `lower` ≤ number of integers in `integers.txt` ≤ `upper`

    (c) `upper` < number of integers in `integers.txt`

50. Write a function definition for a function that will receive

    - a 2D array of `int`s with 1000 columns
    - an `int` representing the number of rows
    - an address of an `int` representing the even count
    - an address of an `int` representing the odd count

    The function should return (via pointers) the count of how many rows of the array have even sums and the count of how many rows have odd sums. Assume that the counts are uninitialized.

51. Write a function definition for a function that will receive

    - a 2D array of `int`s with 1000 columns
    - an `int` representing the number of rows

    The function will return the minimax value (i.e., minimum of the maximums) of the columns of the array.

52. Write a program that can read two files, `first.txt` and `second.txt`, and count how many of the lines in `first.txt` are in `second.txt`. The program should print the final count. Each file has exactly 1000 lines; each line is unique within that file and consists of at most 49 characters. There is no delimiter.

53. Write a program that can read a comma-delimited file, `movies.csv`, storing each line in a structure. Each line consists of

    - title – represented by a string of at most 99 characters
    - category – represented by a string of at most 29 characters
    - release year – represented by a four-digit integer

    The file could consist of as many as 1000 lines. After storing the lines, print them from the array of structures in reverse order, only printing those structures that you stored movie information in.

54. Store the following integers in an array:

```
5, 8, 2, 7, 6, 3, 14, 1, 0, 4
```

Print the running sum of the integers as long as the sum remains less than 50% of the sum of all of the integers.

55. Store the following secret message in a program:

```
char message[] = "S*adem*oek*bel[*fhe]hWcc_d]8";
```

Your program should decrypt the message with the following process:

- If the ASCII value is greater than or equal to 87, add 10 to its value.
- If the ASCII value is less than 87, subtract 10 from its value.

56. Store the following set of XY coordinates in a 2D array:

```
X, Y
====
8, 3
9, 2
6, 6
3, 7
4, 5
0, 8
5, 1
```

Write a program that prompts the user for the X and Y coordinates (as integers) of a new point and then determines which of the coordinates in the set above is closest to the new point. Print the X and Y coordinates of the closest point as well as the distance to it.

To find the distance between two points, $(x_1,\ y_1)$ and $(x_2,\ y_2)$, use the Euclidean distance:

$$\text{distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

57. Store the XY coordinates from the previous problem in a 2D array. Write the logic to determine which two points are farthest from each other.

58. Prompt the user for two integers, a and b, such that $0 < a \leq b$. Then for each integer in the range of a to b (inclusive), determine how many divisors it has and show this graphically by printing a star for each divisor. For example,

```
Enter an integer greater than zero:  3
Enter an integer greater than the first:  8
  3: **
  4: ***
  5: **
  6: ****
  7: **
  8: ****
```

59. We have the following three sets of data:

```
group 1
 X   Y
--------
 0, 11
 1, 15
 5, 10


group 2
 X   Y
--------
10, 3
14, 2
13, 0
12, 1
13, 5


group 3
 X   Y
--------
18, 10
17, 15
14, 16
20, 12
```

Store each of the three groups in its own 2D array. For each group of points, calculate the center of the group as well as the intra-cluster distance (i.e., the average distance) from each point within the group to the center of that group.
Additional requirements:

- To calculate the center of a group, write a function with the following input:
  - a 2D array of `ints` with two columns; this is a group of points

  - an `int`; this is the number of rows of the 2D array
  - a `double*`; this is the mean X value (pointer variable)
  - a `double*`; this is the mean Y value (pointer variable)

  The function will have return type `void` and be used for each group. This function will use pointers to get the mean X and Y values back to `main()`. This will require that these variables be defined in `main()` and their addresses passed to the function during the function call. DO NOT PRINT FROM THIS FUNCTION.

  How do you calculate the center? The mean X value is the sum of the X values divided by the total number of X values. The mean Y value is the sum of the Y values divided by the total number of Y values.

- To calculate the intra-cluster distance, write a function with the following input:
  - a 2D array of `ints` with two columns; this is a group of points
  - an `int`; this is the number of rows of the 2D array
  - a `double`; this is the mean X value
  - a `double`; this is the mean Y value

  The function will have return type `double` and be used for each group. The number and types of the function parameters should match what is given above. DO NOT PRINT FROM THIS FUNCTION.

  How do you calculate the intra-cluster distance? Sum the distances from each point in a group to the center of that group and divide this sum by the total number of points. The distance will be the Euclidean distance as defined in previous assignments.

60. Write a program that prompts the user for a number in word form (e.g., "twenty-seven"); the program will then write out the equivalent integer amount. You only need to deal with cases in the range of 20 to 99, i.e., twenty to ninety-nine, although if you want more of a challenge you should also deal with 1-18. Compound words should be hyphenated, for example, `thirty-six` instead of `thirty six`. Examples:

```
Enter an amount (as a word):  one
You entered 1

Enter an amount (as a word):  eleven
You entered 11

Enter an amount (as a word):  sixteen
You entered 16

Enter an amount (as a word):  thirty-eight
You entered 38

Enter an amount (as a word):  ninety-nine
You entered 99
```

61. Write a program that produces counts for the following string of data: count of lowercase characters, count of uppercase characters, count of punctuation characters, and count of spaces.

    Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

    Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

62. There are many functions withing the string library. Implement your own version of

    - `strlen()`
    - `strcmp()`
    - `strcat()`

    Note that you should give your versions of these functions names that differ from those in the string library.

63. Store a string of your choice in a program that then reads the string, storing the address of the beginning of each word in an array of type `char *`. Note that you will have to preallocate space for the array of type `char *`. Then use this array to print each word and what follows it in the string. For example, if the string is

    ```
    this is my sample string.
    ```

    then the output would be

    ```
    this is my sample string.
    is my sample string.
    my sample string.
    sample string.
    string.
    ```