Assignment#1

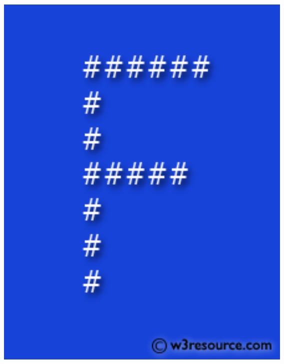
C Basic Declarations and Expressions:

Exercise-1.1 (cut and paste)

Write a C program to print a block F using hash (#), where the F has a height of six characters and width of five and four characters.

C Code:

```
#include <stdio.h>
 int main()
 {
        printf("######\n");
        printf("#\n");
        printf("#\n");
        printf("#####\n");
        printf("#\n");
        printf("#\n");
        printf("#\n");

        return(0);
}
```



```
######
#
#
#####
#
#
#
```

Copy

Sample Output:

```
######
#
#
#####
#
#
#
```

Exercise-1.2 (You Try – on your own)

Modify the code above to **create your first and last initials** using "*" (asterisks)

NOTE: **\n** means (move to next line) **\t** means (tab a few spaces)
Leaving a space in the double quotes of a printf command will leave a space. `printf("Hello world")` *becomes:* Hello world.
THESE special slash \ characters are called **ESCAPE SEQUENCES.** See more on next page.

ESCAPE SEQUENCES.

An escape sequence is a series of characters that represents a special character. It begins with a backslash character (\), which indicates that the character(s) that follow the backslash character should be treated in a special way. C uses escape sequences within a format string to print certain special characters. For example \n moves the output position to the beginning of the next line. The following is a list of escape sequences.

Escape sequence

\n	prints a new line
\b	backs up one character
\t	moves the output position to the next tab stop
\\	prints a backslash
\"	prints a double quote
\'	prints a single quote
\a	make an alert or beep sound

Exercise-1.3 `scanf()` (You type in)

The `scanf()` function allows you to accept input. For now we can only accept input from the keyboard.

This program *takes in* two integers and finds their sum

```
#include <stdio.h>
int main()
   {
     int x, y, sum;
    printf("\nInput the first integer: ");
    scanf("%d", &x);
    printf("\nInput the second integer: ");
    scanf("%d", &y);
    sum = x + y;
    printf("\nSum of the above two integers = %d\n", sum);
    return 0;
   }
```

Sample input: 4 6

Sample output: 10

Exercise-1.4 (You Create and Check) – Solution next page

Create a program that takes in a length and width of a rectangle and then outputs the perimeter and area

Sample input:

4 7

Sample output:

Area: 28 meters squared

Perimeter: 22 meters

(Solution below) only check when you have tried the exercise yourself.

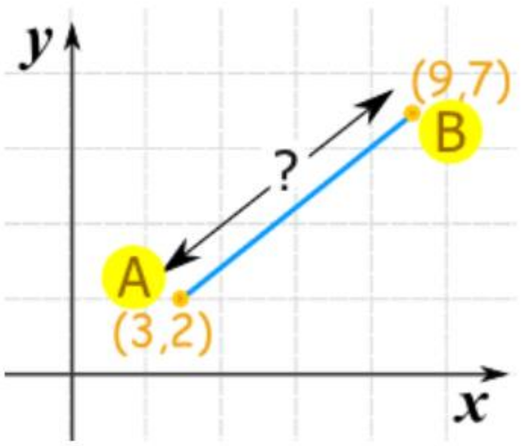
Solutions Exercise 1.4

```
#include <stdio.h>
/* height and width of a rectangle in inches */
int width;
int height;

int area;
int perimeter;

int main()
{
    printf("Give me a height");
    scanf("%d",&height);
    printf("give me a width");
    scanf("%d",&width);

    perimeter = 2*(height + width);
      printf("Perimeter of the rectangle = %d inches\n", perimeter);

      area = height * width;
      printf("Area of the rectangle = %d square inches\n", area);

return(0);
}
```

Exercise 1.5 (*Possibly* try or type it out) – solution on next page. Involves floats.

```
Create a program that can find the distance between any two points

Note: Distance between two points:
```

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



```
Sample Input:

X1 = 3
Y1 = 2

X2 = 9
Y2 = 7

Sample Output:

7.8
```

Exercise 1.5 (solution)

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

int main()
{
	float x1, y1, x2, y2, gdistance;
	printf("Input x1: ");
	scanf("%f", &x1);
	printf("Input y1: ");
	scanf("%f", &y1);
			printf("Input x2: ");
	scanf("%f", &x2);
	printf("Input y2: ");
	scanf("%f", &y2);
	gdistance = ((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1));
	printf("Distance between the said points: %.4f",
sqrt(gdistance));
	printf("\n");
	return 0;
}
```

Exercise 1.6 BOOLEAN and if/else (Just Type it in) – solution on next page

Write a C program that **reads an integer** and checks to see which specified range it fits in. Print an error message if the number is negative and greater than 80. Specified Range: [0, 20], [21, 40], [41, 60], [61, 80]

Sample input: 43
Sample output: Range [41-60]

Exercise 1.6 Solution:

```
#include <stdio.h>
int main()
{
        int x;
        printf("\nInput an intger: ");
        scanf("%d", &x);
        if(x >=0 && x <= 20)
        {
                printf("Range [0, 20]\n");
        }
        else if(x >=21 && x <= 40)
        {
                printf("Range (25,50]\n");
        }
        else if(x >=41 && x <= 60)
        {
                printf("Range (50,75]\n");
        }
        else if(x >61 && x <= 80)
        {
                printf("Range (61,80]\n");
        }
        else
        {
        printf("Outside the range\n");
        }
        return 0;
}
```

Exercise 1.7 While loop (Try and Check)

Suppose that you want to add all the integers from 1 to 1000. If you follow the previous examples, you would require a thousand-line program! Instead, you could use a *loop* in your program to perform a *repetitive* task

Write a program which sums all the integers from 1 to an **upperbound provided by the user,** using a so-called *while-loop*.

Sample input: 455

Sample output: 103740

Solution Exercise 1.7 (try on your own first)

```
#include <stdio.h>

int main()

 {
   int sum = 0;
   int upperbound;

   printf("Enter the upperbound: ");
   scanf("%d", &upperbound);

   int number = 1;
   while (number <= upperbound)

   {
      sum = sum + 1;

   }

   printf("The sum from 1 to %d is %d.\n", upperbound, sum);

   return 0;
}
```

Exercise 1.8 % Modulo and For loop **(Just type in) – Read For loop note first.**

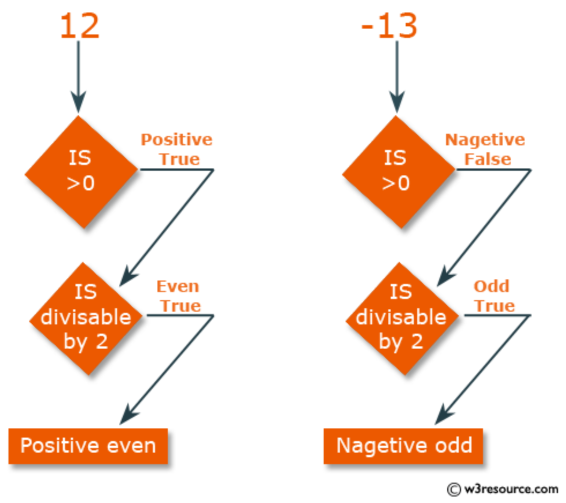
The % operator in C is the “modulo” operator. Its value is the remainder of the integer division of its two operands. For example, if x is 13, then the value of “x % 2” would be 1, since 13 divided by 2 is 6 with remainder 1.

This operation has tons of great applications (one of which is sorting out if an number is odd or even.

Exercise 1.8 (Type in) Solution on next page

Write a C program to check a given integer is *positive even*, *negative even*, *positive odd or negative odd*. Print even if the number is 0.

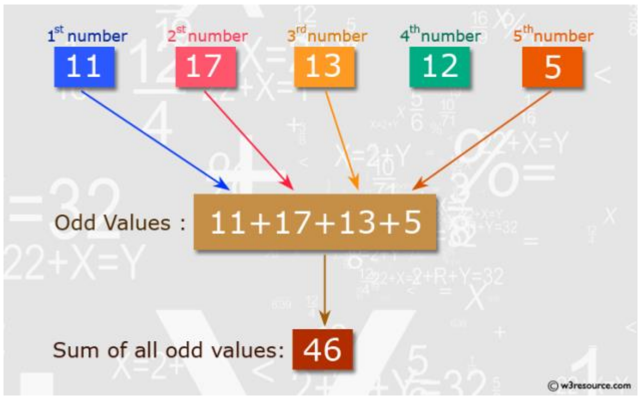
Sample input 12
Sample output: Positive even



Exercise 1.8 Solution

```
#include <stdio.h>
int main()
{
    int x;
     printf("Input an integer: ");
     scanf("%d", &x);

        if(x == 0)
         {
                printf("Positive\n");
          }
          else if(x < 0 && (x%2) != 0)
          {
                printf("Negative Odd\n");
          }
          else if(x < 0 && (x%2) == 0)
          {
                printf("Negative Even\n");
          }
          else if(x > 0 && (x%2) != 0)
          {
                printf("Positive Odd\n");
          }
          else if(x > 0 && (x%2) == 0)
          {
                printf("Positive Even\n");
          }

     return 0;
}
```

Exercise 1.9 (type in)
ARRAYS. Review or Arrays or see MR. Walzl if you need a refresher.
Write a C program that can *read 5 numbers and sum of all odd values between them.*



Exercise 1.9 Solutions

```
#include <stdio.h>
int main()
{
        int j, numbers[5],total=0;
        printf("\nInput the first number: ");
    scanf("%d", &numbers[0]);
    printf("\nInput the second number: ");
    scanf("%d", &numbers[1]);
    printf("\nInput the third number: ");
    scanf("%d", &numbers[2]);
        printf("\nInput the fourth number: ");
    scanf("%d", &numbers[3]);
    printf("\nInput the fifth number: ");
    scanf("%d", &numbers[4]);
        for(j = 0; j < 5; j++)
        {
                if((numbers[j]%2) != 0)
                {
                   total += numbers[j];
                }
        }
        printf("\nSum of all odd values: %d", total);
        printf("\n");
        return 0;
    }
```

Exercise 1.10 ARRAYS (type in)

Several new things going on here:

- **Arrays** – again, review arrays or see Mr. Walzl if you need a refresher.

- The **+=** operator in C is one of the language's *compound assignment* operators. It is essentially a shorthand notation for incrementing the variable on the left by an arbitrary value on the right.

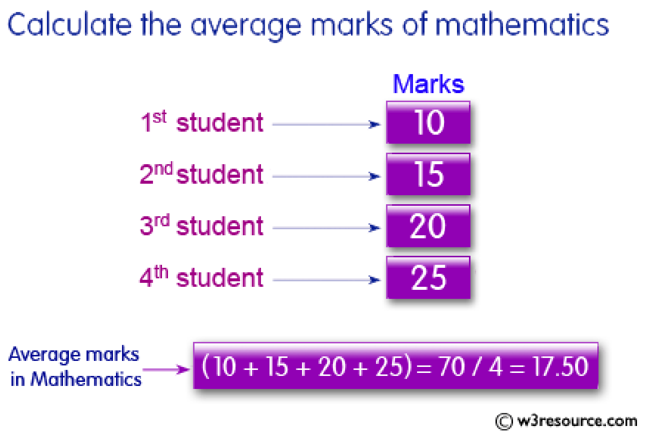
  The following two lines of C code are identical, in terms of their effect on the variable z:

  ```
  1. z = z + y;   // increment z by y
  2. z += y;     // increment z by y
  ```

- **Break** is a statement in C. It can be used either in loops or *switch case*. If used in loop, it exits the loop and runs the rest of the code. i.e. terminates the loop.

- **`float(integer)`** **–** turns a variable into a float ex: **`float(7)=7.0000`**

Write a C program to calculate the average marks of mathematics of some students. Input 0 (excluding to calculate the average) or negative value to terminate the input process.



Sample Input Mathematics marks (0 to terminate): 10
15
20
25
0
Sample Output Average marks in Mathematics: 17.50

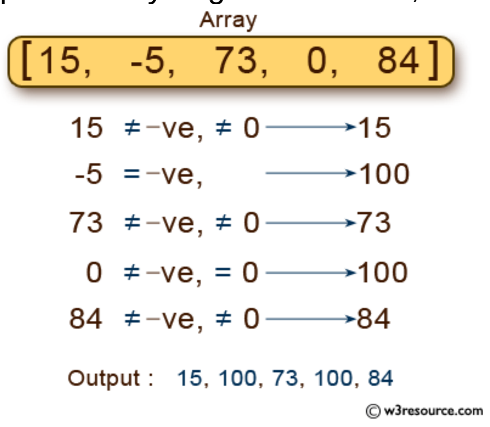
Copy (type in the solution on the next page)

Exercise 1.10 Solution:

```
#include <stdio.h>
int main()
{
        int marks[99], m, i, a=0, total=0;
        float f;
        printf("Input Mathematics marks (0 to terminate): ");
        for(i = 0; ; i++)
        {
                scanf("%d", &marks[i]);
                if(marks[i] <= 0)
                {
                break;
                }
                a++;
                total += marks[i];
        }
        f = (float)total/(float)a;
        printf("Average marks in Mathematics: %.2f\n", f);
        return 0;
}
```

Exercise 1.11 more ARRAYS (TRY or type in)

Write a C program that reads 7 elements of an array then prints out the array but replace every negative number, or zero with 100.



Solution on next page

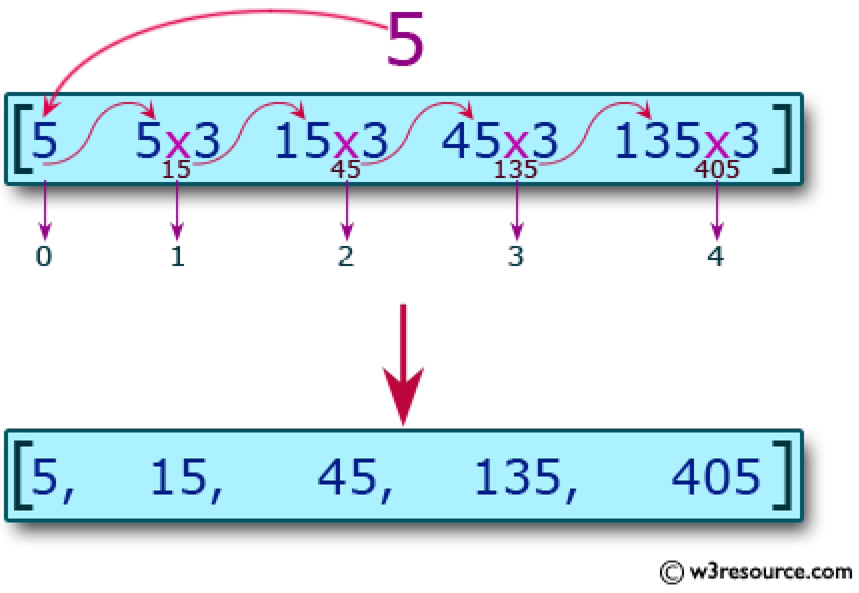
Sample Solution:

```
#include <stdio.h>
int main()
       {
       int n[5], i, x;
       printf("Input the 5 members of the array:\n");
       for(i = 0; i < 5; i++)
       {
              scanf("%d", &x);
              if(x>0)
              {
                n[i] = x;
              }
              else
              {
                     n[i] = 100;
              }
       }
       printf("Array values are: \n");
       for(i = 0; i < 5; i++) {
              printf("n[%d] = %d\n", i, n[i]);
       }
       return 0;
}
```

Exercise 1.12 even more ARRAYS (you try then peek)

Write a C program that reads a number…. and then creates and prints an array starting with the given number and each element after is triple of the previous element. The array must contain 5 elements
For example, if given # is 2, the array numbers must be 2, 6, 18, 54 and 162

Another example:



```
Input the first number of the array:
5
n[0] = 5
n[1] = 15
n[2] = 45
n[3] = 135
n[4] = 405
```

Try on your own…then peek at solution.

Solution on the next page

Sample solution:

```
#include <stdio.h>
int main()
 {
        int n[5], i, x;
        printf("Input the first number of the array:\n");

        scanf("%d", &x);
        for(i = 0; i < 5; i++)
        {
                n[i] = x;
                x = 3*x;
        }

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

Exercise 1.13 (YOU TRY….then peek at solution)

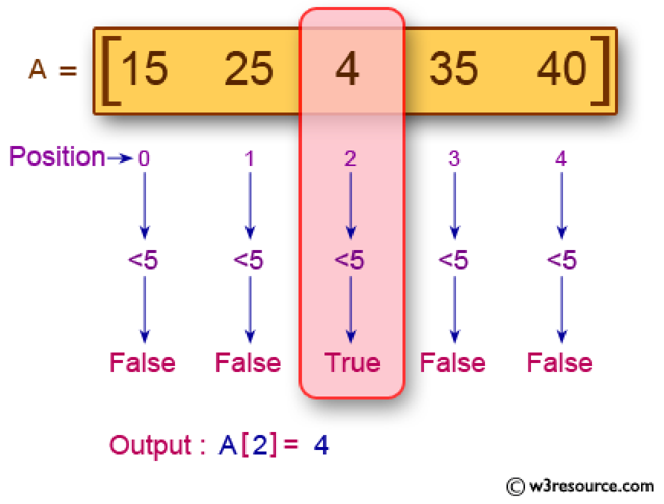
Write a C program to read an array of length 5 and print the position and value of any array elements of value less than 5.

Sample Input:

15
25
4
35
40

Sample output:

A[2] = 4.0



Sample Solution:

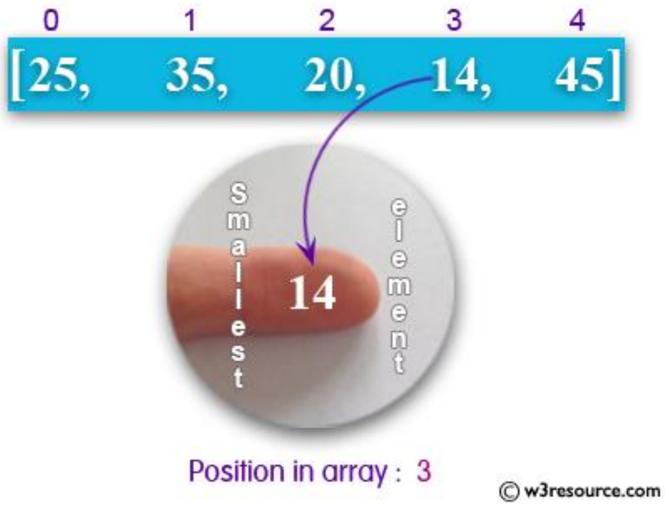
```
#include <stdio.h>
#define AL 5
#define MAX 5

int main() {
        float N[AL];
        int i;
        printf("Input the 5 members of the array:\n");
        for(i = 0; i < AL; i++) {
                scanf("%f", &N[i]);
        }
        for(i = 0; i < AL; i++) {
                if(N[i] < MAX) {
                        printf("A[%d] = %.1f\n", i, N[i]);
                }
        }
        return 0;
}
```

Exercise 1.14 (you type)

Write a C program to read an array of length 6 and find the smallest element and its position.

Sample Input the array elements:
25
35
20
14
45

Sample Output
Smallest Value: 14
Position of the element: 3



Sample Solution:

```c
#include <stdio.h>
int main()
{
	int e, i, sval, position;

	printf("\nInput the length of the array: ");
	scanf("%d", &e);

	int v[e];
	printf("\nInput the array elements:\n ");

	for(i = 0; i < e; i++)
	{
			scanf("%d", &v[i]);
	}
	sval = v[0];
	position = 0;

	for(i = 0; i < e; i++)
	{
		if(sval > v[i])
		{
			sval = v[i];
			position = i;
		}
	}

	printf("Smallest Value: %d\n", sval);
	printf("Position of the element: %d\n", position);
	return 0;
}
```

Exercise 1.15 (random number generation and “do while” loop) YOU TYPE

Write a C program to generate a random number between 1 and 10... and then gets user to guess the number until they get it correct.

Sample Input: 7
Sample Output: Number is higher. Guess again

Notice that this code uses a **“do while”** loop. Same as a while loop except the decision is after the code that get executed. **This loop is used if code must be executed AT LEAST ONCE**.

Sample Solution:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

        int main ()
        {
                int number, input;       /* initialize random seed: */
                srand ( time(NULL) );   /* Generate a random number: */
                number = rand() % 10 + 1;
                do      {
                                printf ("\nGuess the number (1 to 10): ");
                                scanf ("%d",&input);
                                if (number > input)
                                        printf ("The number is higher\n");
                        } while (number!=input);
                printf ("That is correct!\n\n");
                return 0;
        }
```