**Explanation of the code:**

# Include Directives:

Both “stdio.h” and “stdlib.h” are there by default.  
“math.h” is there for math functions such as pow and sqrt (See function “**one”** below)  
“time.h” is there for a better randomization. (See function “**five”** below)  
“locale.h” is there for language support. (See function “**six”** below)

# Function Prototypes: They're needed for functions to be identified by the compiler.

**Functions:**

int **main**():   
It simply calls the function “**menu”** and returns 0 at the end.

int **menu**():   
It prints the menu to the screen, asks to user what his/her selection is, calls the function “**operations”** with user'sselectionand returns 0 at the end.

int **operations**(int **selection**):  
It takes user's selection from the “**menu”** function and do the corresponding action\* and returns 0 at the end.

\*:

If user's selection is **0,** it calls the function “**zero”** and **break**sthe switch case at the end.

If user's selection is **1,** itcalls the function “**one”** and returns to the **menu** at the end.

If user's selection is **2,** itcalls the function “**two”** and returns to the **menu** at the end.

If user's selection is **3,** itempties the buffer (if buffer isn't emptied here, **scanf** in the function “**three”** does not work properly),prints “Write a sentence:” to the screen, calls the function “**three”** and returns to the **menu** at the end.

If user's selection is **4,** itcalls the function “**four”** and returns to the **menu** at the end.

If user's selection is **5,** itcalls the function “**five”** and returns to the **menu** at the end.

If user's selection is **6,** itcalls the function “**six”** and returns to the **menu** at the end.

If user's selection is **not** between **0** and **6,** it prints a warning message and returns to the “**menu”** function.

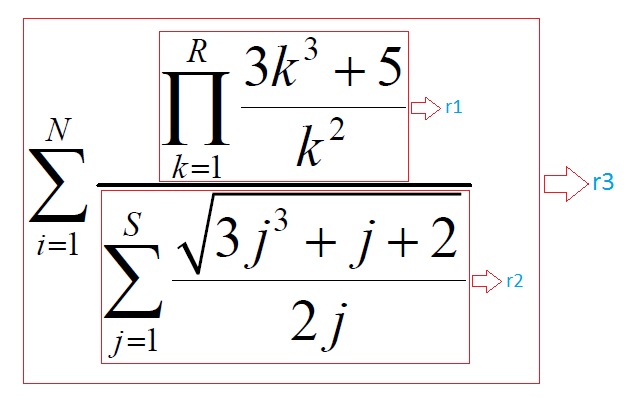
int **zero**()**:**It asks to user whether he is sure or not about exiting.   
If the user enters “Y” or “y” it returns 0 and the program ends.  
If the user enters “N” or “n” it returns to the **menu.**If the user enters something different than “Y”, “y”, “N” or “n” it prints a warning massage and returns to the function **zero**.

**Important Note**: If the “confirmation” variable is tried to be taken by “%c” rather than “ %c” the program doesn't wait user to enter anything at first and reads some value from somewhere. Probably a buffer issue, again.

There was two ways to solve this:   
1) Emptying the buffer.   
2) Leaving a space before %c.

I used the second solution for this problem. But please note that this 'trick' doesn't work for every buffer problem, so I had to empty buffer in “**case 3”** above.

void **one**():  
It asks user to enter a value for N. If user enters a non-positive number it prints a warning massage and asks user to enter a new value for N, until user enters a positive number. Same steps are applied for getting a positive value for R and S from the user.  
When the value taking is done, program divides the math equation into 3 main parts and solve them separately. Here is a representation of it:



r1 and r2 are calculated with **for** loops.

r1's initial value is 1. Because 1 is the neutral element of multiplication.

r2's initial value is 0. Because 0 is the neutral element of addition.

Neither r1 nor r2 section of the problem have “i” variable in them. Because of this r3 equals to N times r1/r2.

And finally program prints the result (i.e. r3) to the screen.

void **two**():  
It asks user to enter a value for the lines of the diamond. If the value that user entered is not odd or is smaller than 3, program gives an appropriate warning and asks user to enter a new value until user enters a proper value for the lines of the diamond.

Program divides the problem into two parts in order to solve it easily:  
1)Printing a (n) lined equilateral triangle  
2)Printing a (n-1) lined reversed equilateral triangle

Please note that the value we get from the user is (2n-1).

void **three**():  
It takes a sentence from the user and reads it character by character with the help of recursion until program sees “\n” (i.e. new line) -that's when the sentence ends- and prints “Reversed version of the inputs:” to the screen. After that, it starts to write the characters from last to the first.

void **four**:  
Program asks user to enter a positive number. If user enters a non-positive number or try to exit before entering at least two positive numbers, it gives an appropriate warning and asks user to enter a new value. When user enters at least two positive numbers and enters -1, 'value taking' step ends. It calls “**four\_Second”** and prints the result to the screen.

**Important Note: “**-1” is also registered into the array. But since it's unwanted I subtracted 1 from “a” in order to get rid of the slot that has -1 in it.

int **four\_Second**(int **y[]**, int **x**):

This function is also known as “**Second”**. Because, well, I don't want to disrupt the sequence of the names of the functions by naming this simply **Second**.

It takes the array and the size of the array (see the explanation about the ‘size of the array’ at function **four**) from the function **four**. And firstly finds the **smallest** number and tries to assign a number that is not the smallest to the **second smallest**. Secondly it tries to find the smallest number that is bigger than the **smallest**.

If there's no second smallest number (if user enters the same numbers only, there wouldn't be any second smallest number for example) it gives a warning and returns to menu.

If there's a second smallest number, it returns its value to the function **four.**

void **five**():  
It randomise the randomisation with “srand(time(Null))” first. After that, it starts to assign random numbers between 1-20 to the array. If the same value already exists within the array, it will disregard that number and will randomly create new numbers till it finds a number which is not in the array.

void **six**():  
It writes the names of the creators of the project**.**