Mandatory Homework #02

#1: Implement function with prototype “double calc_pow(unsigned int n);”
that takes as input n and returns (without using pow or similar functions) 2 raised to the power n ($2n$, for
example $2^4 = 2 * 2 * 2 * 2 = 16$). Use while loop to calculate the result. Use double data type for your
calculations. Handle also $n = 0$.
Call the function calc_pow from main function with arguments 4, 1023, and 1025.
a) Verify that calc_pow(4) is indeed 16 (the correct answer)
b) Display the result to calc_pow(1023). Confirm that its correct. Hint: use %g for displaying double for
more practical output.
c) What is the result of calc_pow(1025)? Explain with details what has happened? What is this behaviour
called? What would happen with integer datatypes in similar situation?

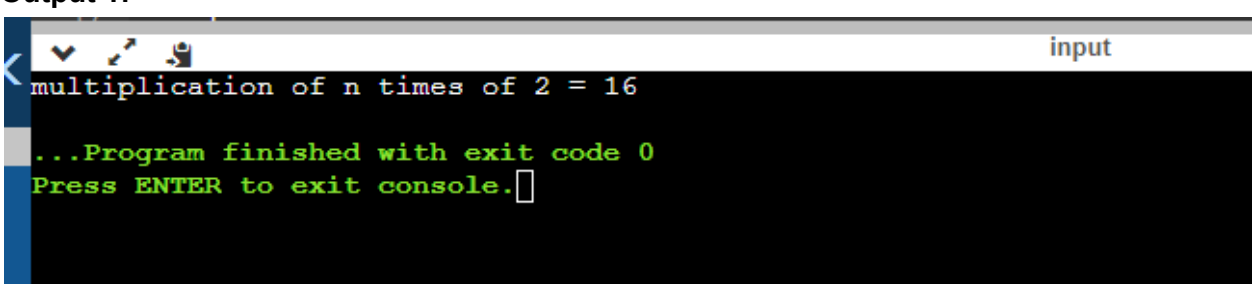
a. **Answer:**

I have mentioned here code for calc_pow (4) with detailed comments in every line. We just put the function prototype because the calc_pow() are behind the main function body. That is why first I need to introduce the clac_pow () function as a prototype. When I called the function I got the correct result as I observed. The output screenshot is attached below. Here, I have used two integer variables namely mult and result , but it should be double, I have checked using double I got the same answer. Because the input value is 4 , so small , that is why it does not make any difference. But It will be different for b and c. Otherwise; we will not get correct answers for b and c.

```
#include <stdio.h>

double calc_pow(unsigned int n);                    //function prototype call here because it
//below the main function

int main()                                          // main function
{
   calc_pow(4); // function call with given number

}
```

```
double calc_pow(unsigned int n)                    // function make here with one input value
{
   int i= 1;                                       // initialize parameter for making loop
   int mult = 2;
   int result;
   if (n == 0)                                     // if input zero then print it 1.
   {
      printf("result for zero = %d", 1);
      return 0;
   }
   else                                            // if not input 0.
   {
      while(i < n)                  //condition will run until given input, <, because i start with 1.
      {
         result = 2 * mult;                 // here 2 multiply n times
         mult = result;                     // result updated in mult
         i++;                               // increment i with 1 until the condition invalid

      }
      printf("multiplication of n times of 2 = %d", mult );        // here print the result

      return mult;                                                 // return the result to main function

   }

}
```

Output 1:

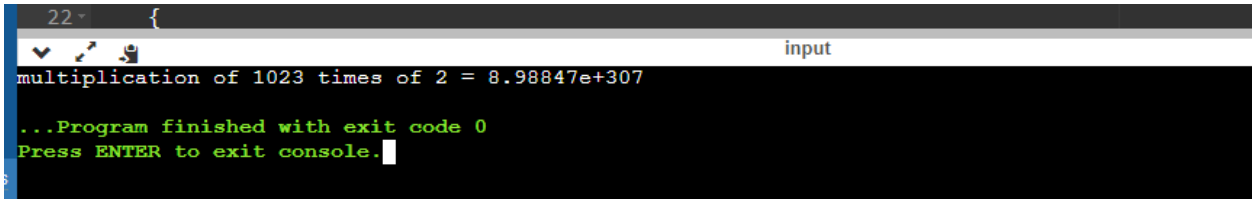


b) Display the result to calc_pow(1023). Confirm that its correct. Hint: use %g for displaying double for more practical output.

Answer:

Here , I have made clear comments in the code scripts for every line. Everything is almost the same as 1 (a) but I have declared all variables as a double to get the correct answer because there are too big numbers like 1023. After calling the calc_pow (1023) ,I got the correct output and I checked it. The code with clear comments and output screenshot are below.

```
#include <stdio.h>

double calc_pow(unsigned int n); //function prototype call here becasue function declare below
main funcion

int main()   // main function
{
   calc_pow(1023); // function call with given number

}

double calc_pow(unsigned int n)   // functiom make here with one inout value
{
   int i= 1;                    // initialize parameter for making loop
   double mult = 2;             // double varibale declare because we want to send double
   double result;               // double varibale declare because we want to send double
   if (n == 0)   // if input zero then print it 1.
   {
      printf("result for zero = %d", 1);
      return 0;
   }
   else            // if not input 1.
   {
      while(i < n)    //condition will run untill given input, <, because i start with 1.
      {
         result = 2 * mult;     // here 2 multiply n times
         mult = result;         // result updated in mult
         i++;                   // increment i with 1 until the condition invalid

      }
      printf("multiplication of 1023 times of 2 = %g", mult );   // here print the result, %g for double
value

      return mult;    // return the result to main function

   }

}
```



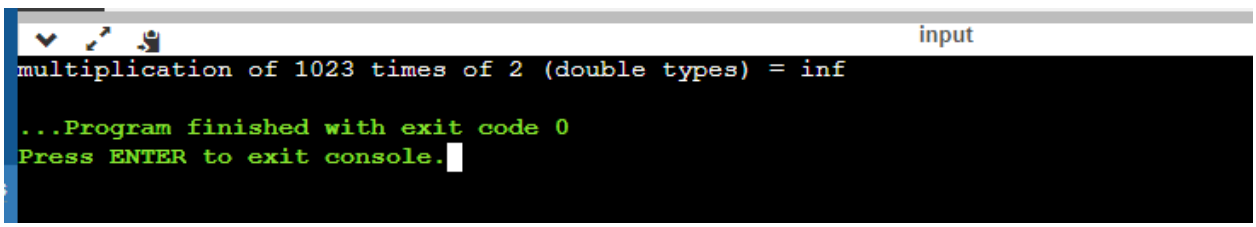
c) What is the result of calc_pow(1025)? Explain with details what has happened? What is this behaviour called? What would happen with integer data types in a similar situation?

Answer:

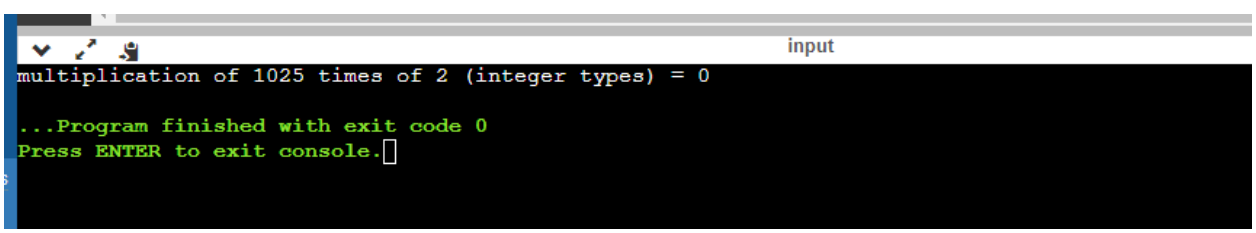
When I called calc_pow(1025), it showed the output infinity. For the double types, this behavior is known as undefined (overflow). Overflow occurs when the number is too large. It represents infinity. The details in terms of coding are below with clear comments associated with the output screenshot.

```
#include <stdio.h>

double calc_pow(unsigned int n); //function prototype call here becasue function declare below main funcion

int main()   // main function
{
   calc_pow(1025); // function call with given number

}

double calc_pow(unsigned int n)   // functiom make here with one inout value
{
   int i= 1;                    // initialize parameter for making loop
   double mult = 2;             // double varibale declare because we want to send double
   double result;               // double varibale declare because we want to send double
   if (n == 0)   // if input zero then print it 1.
   {
      printf("result for zero = %d", 1);
      return 0;
   }
   else            // if not input 1.
   {
```

```
    while(i < n)    //condition will run untill given input, <, because i start with 1.
    {
        result = 2 * mult;     // here 2 multiply n times
        mult = result;         // result updated in mult
        i++;                   // increment i with 1 until the condition invalid

    }
    printf("multiplication of 1023 times of 2 (double types) = %g", mult );   // here print
the result, %g for double value

    return mult;    // return the result to main function

  }

}
```



When I put the integer data types instead of doubles I got this kind of output Like 0. It represents the under flow . it occurs when data is too small to represent. It represents 0.



#2: What is the difference between logical and bitwise operations?

Answer:
The difference between logical and bitwise operation is that logical operation makes decisions based on the multiple operation of ground truth value while bitwise operation works on bits and

performs bit by bit. Such as , for logical operator for And logic is, 0 && 5 = False , and for 5 || 5 = true. On the other hand, bitwise operations first need to convert in binary numbers for both values like 5 and 5, then need to make operations between bit to bit. This calculation is given in detail in question a and b.

a) Calculate by hand (show the calculations in the report) and by C code bitwise AND between decimal numbers 14 and 5.

Answer:

Here : 14 = 1 1 1 0
5 = 0 1 0 1
14 & 5 = 0 1 0 0

Then, the result is , 14 & 5 = 4.

Here , c code output

main.c

```c
/*****************************************************************************

                                Online C Compiler.
                    Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*****************************************************************************/

#include <stdio.h>

int main()
{

    printf("The bitwise operation result between 14 & 5 = %d", 14 & 5 );

    return 0;
}
```

input

```
The bitwise operation result between 14 & 5 = 4

...Program finished with exit code 0
Press ENTER to exit console.
```

b) What is logical AND between 14 and 5?

Answer: For logical operation , it works with truth value.

Here, 14 and 5 is the truth value and we can consider it 1. So two truth values and logical operation should be true.
14 && 5 = true. And c code true presents as 1.

C_code Out put:

main.c

```c
/*****************************************************************************

                                Online C Compiler.
                    Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*****************************************************************************/

#include <stdio.h>

int main()
{

    printf("The logical operation result between 14 && 5 = %d", 14 && 5 );

    return 0;
}
```

input

```
The logical operation result between 14 && 5 = 1

...Program finished with exit code 0
Press ENTER to exit console.
```

#3: Explain in your report step by step why 10+4%5*3/5 is exactly 12.

Answer:

According to the operator precedence and associativity,
The priority is in the following way: %, *, /, and +.
So , 10 + 4% 5 * 3 /5
=10 + 4*3/5
= 10 + 12/5
= 10 + 2 (it takes 2 in code)
= 12

When I calculated step by step by hand according to the operator precedence and associativity rules, I got the answer 12. But When I made the c code for this , I observed the result with 12.

C code output :

The result for the above equation is 12 in c code.

main.c

```c
/*****************************************************************************

                    Online C Compiler.
            Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*****************************************************************************/

#include <stdio.h>

int main()
{

    printf("The calcuöation in c = %d", 10+4%5*3/5 );

    return 0;
}

```

input

```
The calcuöation in c = 12

...Program finished with exit code 0
Press ENTER to exit console.
```