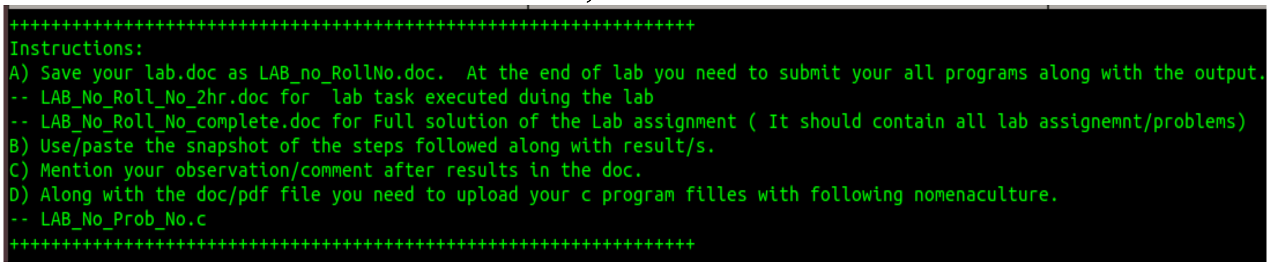
CSB101: Problem Solving and Computer Programmin

Daksh Verma
231210036

LAB 3: Constant, Variables and Data



```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Instructions:
A) Save your lab.doc as LAB_no_RollNo.doc.  At the end of lab you need to submit your all programs along with the output.
-- LAB_No_Roll_No_2hr.doc for  lab task executed duing the lab
-- LAB_No_Roll_No_complete.doc for Full solution of the Lab assignment ( It should contain all lab assignemnt/problems)
B) Use/paste the snapshot of the steps followed along with result/s.
C) Mention your observation/comment after results in the doc.
D) Along with the doc/pdf file you need to upload your c program filles with following nomenaculture.
-- LAB_No_Prob_No.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Objective(s):

- To be familiar with Constant, variables and different data types in C.

++

LAB Exercise

PART A : Conceptual Questions

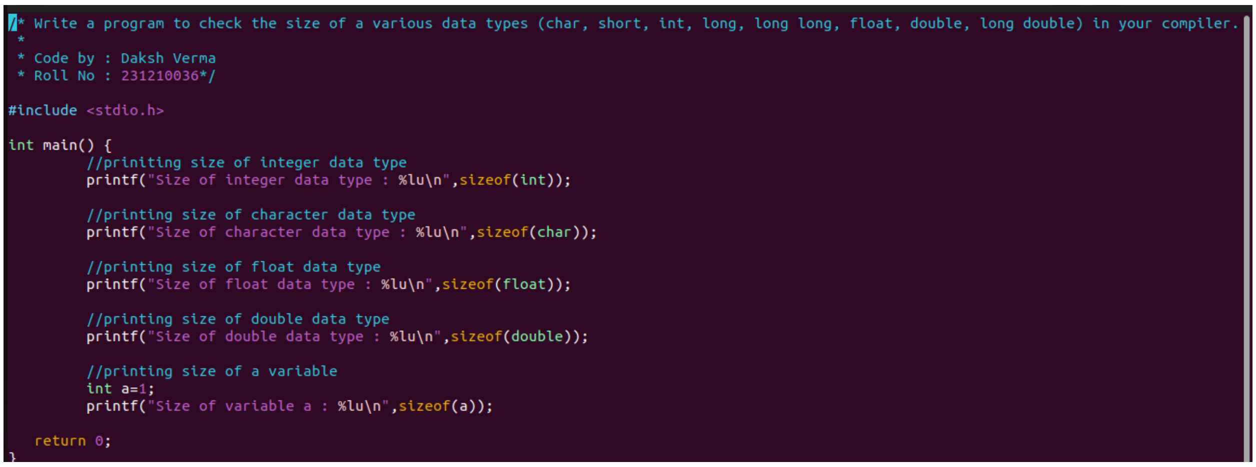
1. Different data types also have different ranges up to which they can store numbers. These ranges may vary from compiler to compiler. Write a program to check the sizeof a various data types (char, short, int, long, long long, float, double, long double) inyour compiler. (**Hint**: *Explore an operator for finding the size*)

Problem Analysis:

The problem is to create a C program that checks and prints the size (in bytes) of various data types, including char, short, int, long, long long, float, double, and long double. To achieve this, we can use the **sizeof** operator, which provides the size of a data type in bytes.

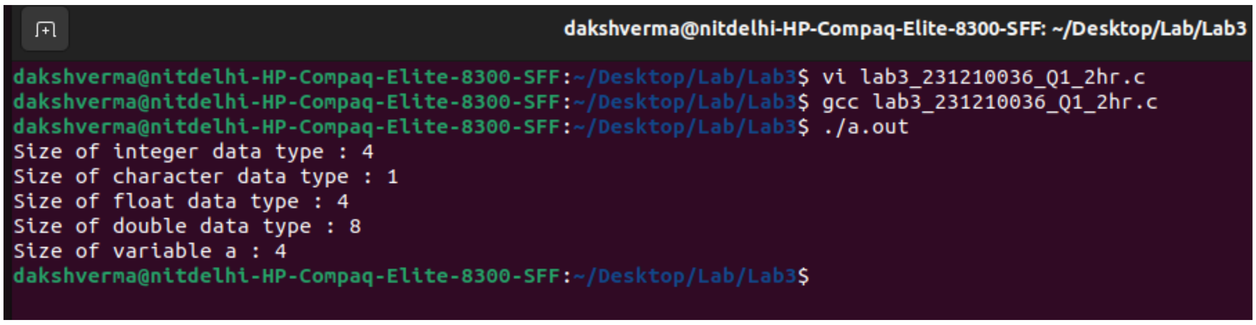
Input Variable	**Processing variable/ Calculations**	**Output variables**	**Necessary header files/functions/macros**
none	A (char)	none	Stdio.h printf()

Code



```c
/* Write a program to check the size of a various data types (char, short, int, long, long long, float, double, long double) in your compiler.
 *
 * Code by : Daksh Verma
 * Roll No : 231210036*/

#include <stdio.h>

int main() {
         //priniting size of integer data type
         printf("Size of integer data type : %lu\n",sizeof(int));

         //printing size of character data type
         printf("Size of character data type : %lu\n",sizeof(char));

         //printing size of float data type
         printf("Size of float data type : %lu\n",sizeof(float));

         //printing size of double data type
         printf("Size of double data type : %lu\n",sizeof(double));

         //printing size of a variable
         int a=1;
         printf("Size of variable a : %lu\n",sizeof(a));

   return 0;
}
```

Output



```
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF: ~/Desktop/Lab/Lab3
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ vi lab3_231210036_Q1_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ gcc lab3_231210036_Q1_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ ./a.out
Size of integer data type : 4
Size of character data type : 1
Size of float data type : 4
Size of double data type : 8
Size of variable a : 4
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$
```

Algorithm:

1. Start
2. Use the **sizeof** operator to determine and print the size of each data type.
    - For char, use **sizeof(char)**
    - For int, use **sizeof(int)**
    - For float, use **sizeof(float)**
    - For double, use **sizeof(double)**
    - For variable, use **sizeof(<variable_name>)**
3. Display the sizes of each data type.
4. Stop

Discussion and Conclusion:

This C program effectively uses the **sizeof** operator to determine and display the size (in bytes) of various data types supported by your compiler. It's a valuable exercise for understanding how different data types occupy memory space in a C program and can help you ensure efficient memory usage in your code.

CSB101: Problem Solving and Computer Programmin

Daksh Verma
231210036

LAB 3: Constant, Variables and Data

2. Write a program to declare two integer and one float variables then initialize them to 19,20, and 12.6666. Also print the variable values in the screen. Explore the different options available in *printf* as discussed in class

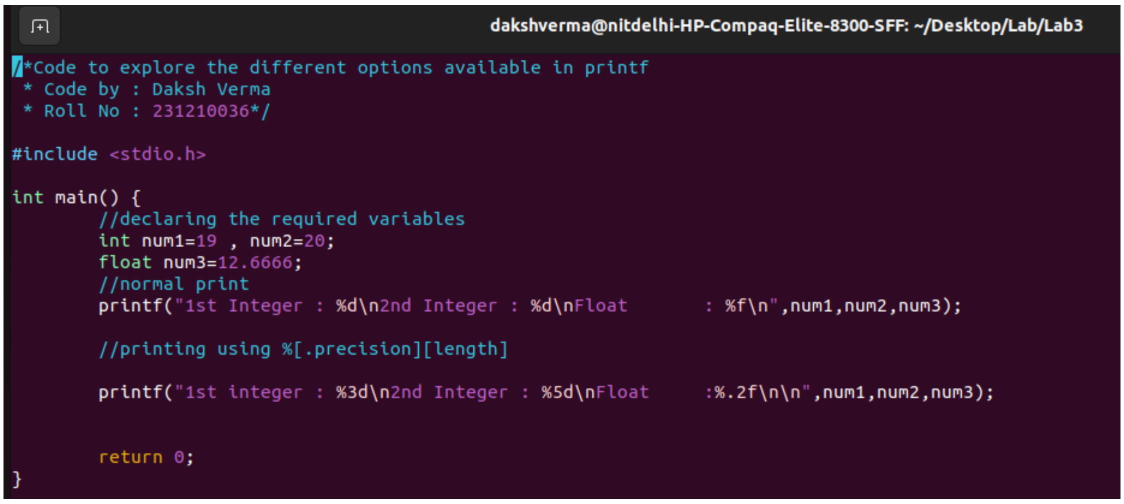
(***Hint***: *Formatted Output*, ***%[flags][width][.precision][length]specifier***).

Problem Analysis:
The problem is to create a C program that declares and initializes two integer variables and one float variable with specific values and then prints these variable values to the screen using the **printf** function. We will explore different formatting options available in **printf** to control the display of these values.

Input Variable	**Processing variable/ Calculations**	**Output variables**	**Necessary header files/functions/macros**
Num1,num2(int) Num3 (float)	none	none	Stdio.h printf()

Algorithm:
1. Start
2. Declare two integer variables, **num1** and num**2**, and one float variable, **num3**.
3. Initialize **num1** to 19, **num2** to 20, and **num3** to 12.6666.
4. Use the **printf** function to print the values of these variables with different formatting options:
   a. Print **num1** using the **%d** specifier.
   b. Print **num2** using the **%d** specifier.
   c. Print **num3** using the **%f** specifier.
   d. Print **num3** with a limited precision of 2 decimal places using the **%.2f** specifier.
   e. Print **num1** and **num1** together using the **%d** specifier and a space in between.
   f. Print **num3** with a width of 10 characters and a precision of 2 decimal places using the **%10.2f** specifier.
5. Stop

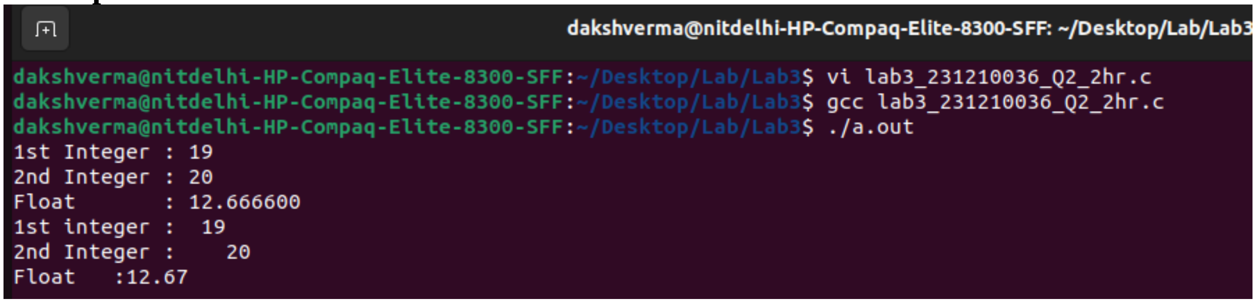
Code



dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF: ~/Desktop/Lab/Lab3

```c
/*Code to explore the different options available in printf
 * Code by : Daksh Verma
 * Roll No : 231210036*/

#include <stdio.h>

int main() {
        //declaring the required variables
        int num1=19 , num2=20;
        float num3=12.6666;
        //normal print
        printf("1st Integer : %d\n2nd Integer : %d\nFloat       : %f\n",num1,num2,num3);

        //printing using %[.precision][length]

        printf("1st integer : %3d\n2nd Integer : %5d\nFloat     :%.2f\n\n",num1,num2,num3);


        return 0;
}
```

Output



```
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF: ~/Desktop/Lab/Lab3
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ vi lab3_231210036_Q2_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ gcc lab3_231210036_Q2_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ ./a.out
1st Integer : 19
2nd Integer : 20
Float       : 12.666600
1st integer :  19
2nd Integer :    20
Float   :12.67
```

Discussion and Conclusion: This C program demonstrates how to declare, initialize, and print integer and float variables using the **printf** function with different formatting options. It showcases the flexibility of the **printf** function in controlling the appearance of data on the screen, including formatting integers, floats, and combining multiple variables in a single output statement. Understanding these formatting options is crucial for customizing the display of data in C programs.

Daksh Verma
231210036

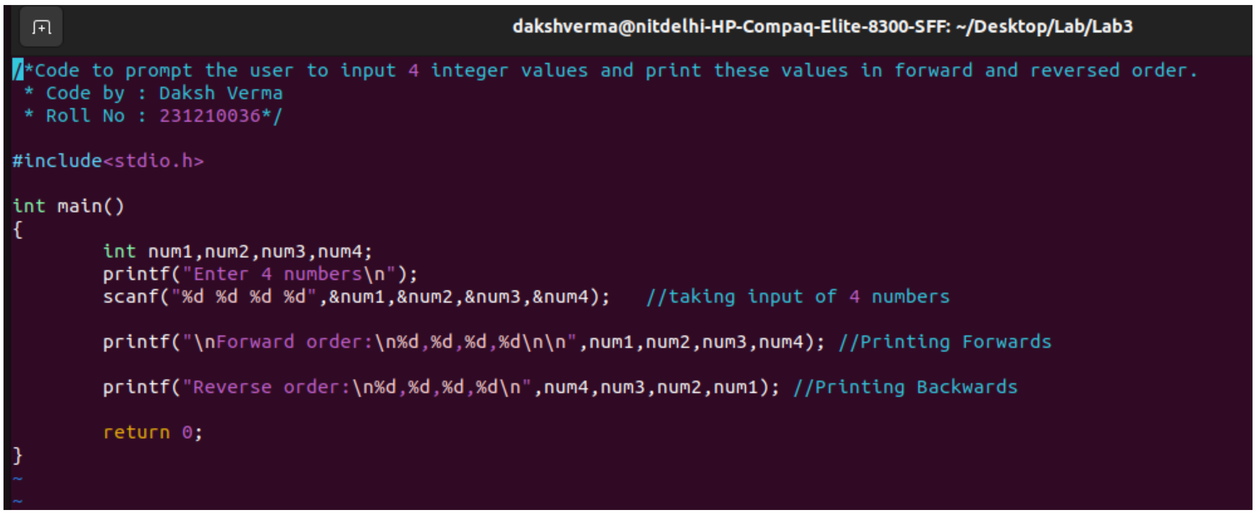
3. Write a C program to prompt the user to input 4 integer values and print these values in forward and reversed order.

Problem Analysis:
The problem requires us to create a C program that interacts with the user, prompting them to input four integer values. Once the values are input, the program should print them in both forward (as entered) and reversed order.

Input Variable	**Processing variable/ Calculations**	**Output variables**	**Necessary header files/functions/macros**
Num1,num2,num3,num4 (int)	none	none	Stdio.h printf()

Algorithm:
1. Start
2. Declare 4 integer variables num1,num2,num3,num4.
3. Display a prompt asking the user to input four integer values
4. Print a header message indicating "Values in forward order."
5. Print the values in forward order (num1,num2,num3,num4)
6. Print a header message indicating "Values in reversed order."
7. Print the values in reverse order (num4,num3,num2,num1)
8. Stop

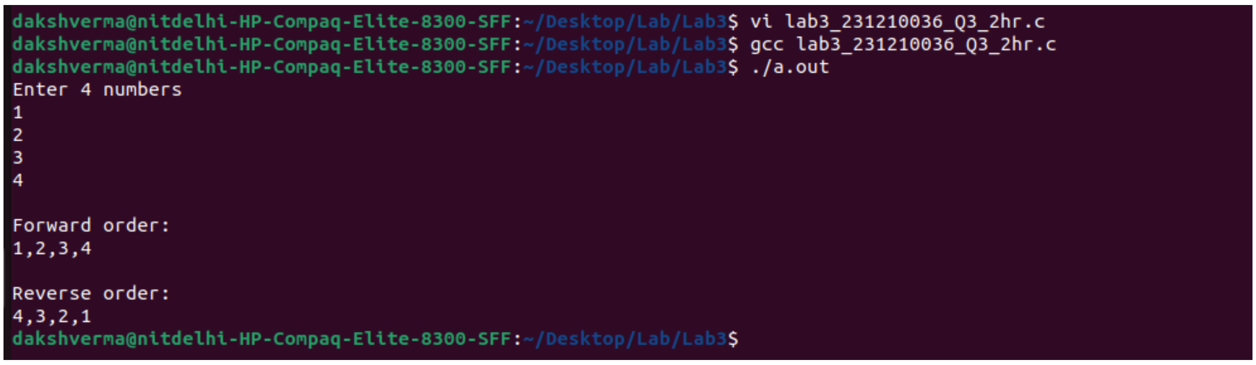
Code


dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF: ~/Desktop/Lab/Lab3

```
/*Code to prompt the user to input 4 integer values and print these values in forward and reversed order.
 * Code by : Daksh Verma
 * Roll No : 231210036*/

#include<stdio.h>

int main()
{
        int num1,num2,num3,num4;
        printf("Enter 4 numbers\n");
        scanf("%d %d %d %d",&num1,&num2,&num3,&num4);   //taking input of 4 numbers

        printf("\nForward order:\n%d,%d,%d,%d\n\n",num1,num2,num3,num4); //Printing Forwards

        printf("Reverse order:\n%d,%d,%d,%d\n",num4,num3,num2,num1); //Printing Backwards

        return 0;
}
~
~
```

Daksh Verma
231210036

Output



```
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ vi lab3_231210036_Q3_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ gcc lab3_231210036_Q3_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ ./a.out
Enter 4 numbers
1
2
3
4

Forward order:
1,2,3,4

Reverse order:
4,3,2,1
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$
```

Discussion and Conclusion:

This C program effectively fulfills the problem requirements by prompting the user for four integer inputs and then printing those values both in forward and reversed order. It provides a practical example of user interaction, input handling.

Daksh Verma
231210036

4. Write a program to swap two variables values with and without using third variables.

Problem Analysis:

The problem is to write a C program that swaps the values of two variables, both with and without using a third variable. For swapping without a third variable, we will need to find a mathematic algorithm to swap the values. We can do this via addition subtraction or multiplication division method.

Input Variable	**Processing variable/ Calculations**	**Output variables**	**Necessary header files/functions/macros**
a , b(int)	Temp (int)	none	Stdio.h printf()

Algorithm for Swapping with a Third Variable:

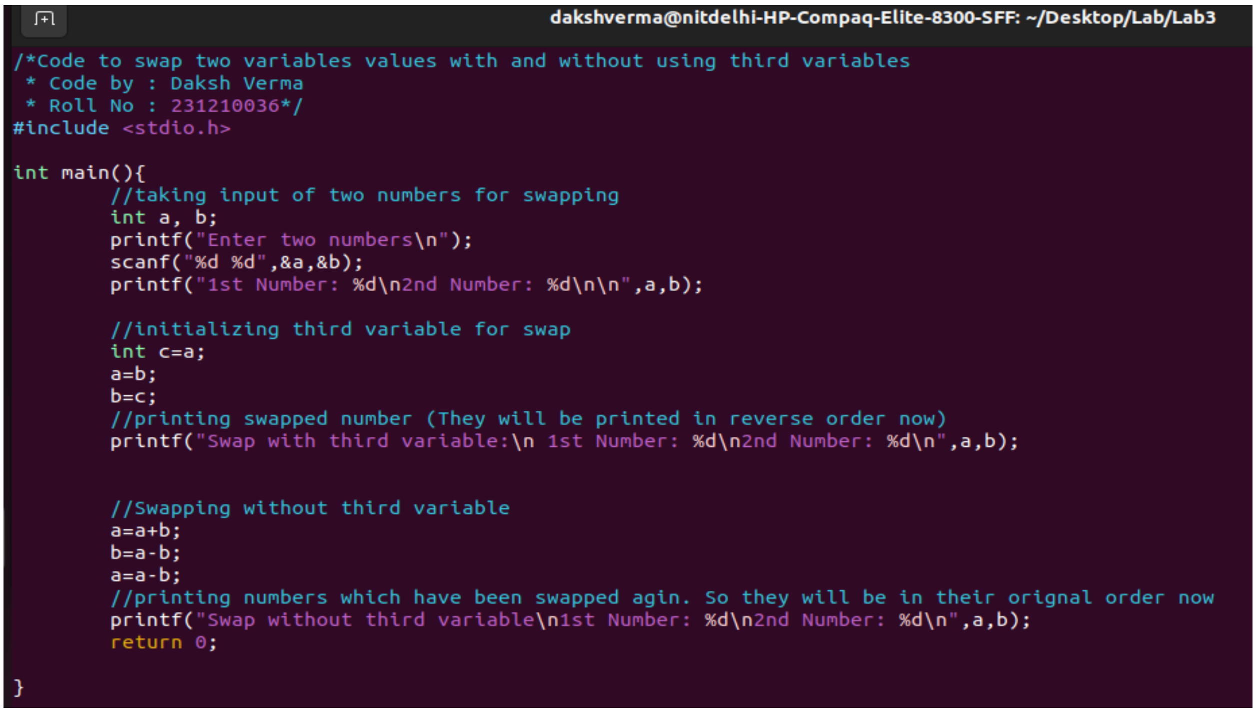
1. Start
2. Declare three integer variables: **a**, **b**, and **temp**.
3. Initialize **a** and **b** with the values to be swapped.
4. Assign the value of **a** to **temp**.
5. Assign the value of **b** to **a**.
6. Assign the value of **temp** to **b**.
7. Display the swapped values of **a** and **b**.
8. Perform the swapping without using a third variable:

   a. Assign **a** the value of **a + b**.

   b. Assign **b** the value of **a - b**.

   c. Assign **a** the value of **a – b**
9. Display the swapped values of **a** and **b**.
10. Stop

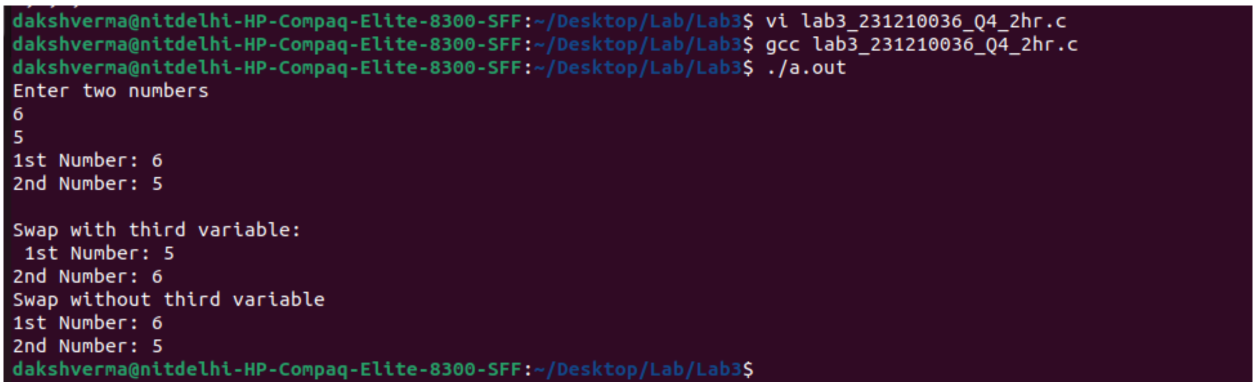
Code



dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF: ~/Desktop/Lab/Lab3

```c
/*Code to swap two variables values with and without using third variables
 * Code by : Daksh Verma
 * Roll No : 231210036*/
#include <stdio.h>

int main(){
        //taking input of two numbers for swapping
        int a, b;
        printf("Enter two numbers\n");
        scanf("%d %d",&a,&b);
        printf("1st Number: %d\n2nd Number: %d\n\n",a,b);

        //initializing third variable for swap
        int c=a;
        a=b;
        b=c;
        //printing swapped number (They will be printed in reverse order now)
        printf("Swap with third variable:\n 1st Number: %d\n2nd Number: %d\n",a,b);


        //Swapping without third variable
        a=a+b;
        b=a-b;
        a=a-b;
        //printing numbers which have been swapped agin. So they will be in their orignal order now
        printf("Swap without third variable\n1st Number: %d\n2nd Number: %d\n",a,b);
        return 0;

}
```

Output



```
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ vi lab3_231210036_Q4_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ gcc lab3_231210036_Q4_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ ./a.out
Enter two numbers
6
5
1st Number: 6
2nd Number: 5

Swap with third variable:
 1st Number: 5
2nd Number: 6
Swap without third variable
1st Number: 6
2nd Number: 5
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$
```

Discussion and Conclusion:

These C programs demonstrate two different methods for swapping the values of two variables. The first method uses a third variable **temp** for temporary storage, while the second method swaps the values without using a third variable by performing arithmetic operations. Both methods provide an understanding of variable manipulation in C programming.

5. Write a program in C to take your name as input and Print " **Hello X, Welcome to NIT Delhi!!**", where X is your name.

Problem Analysis:

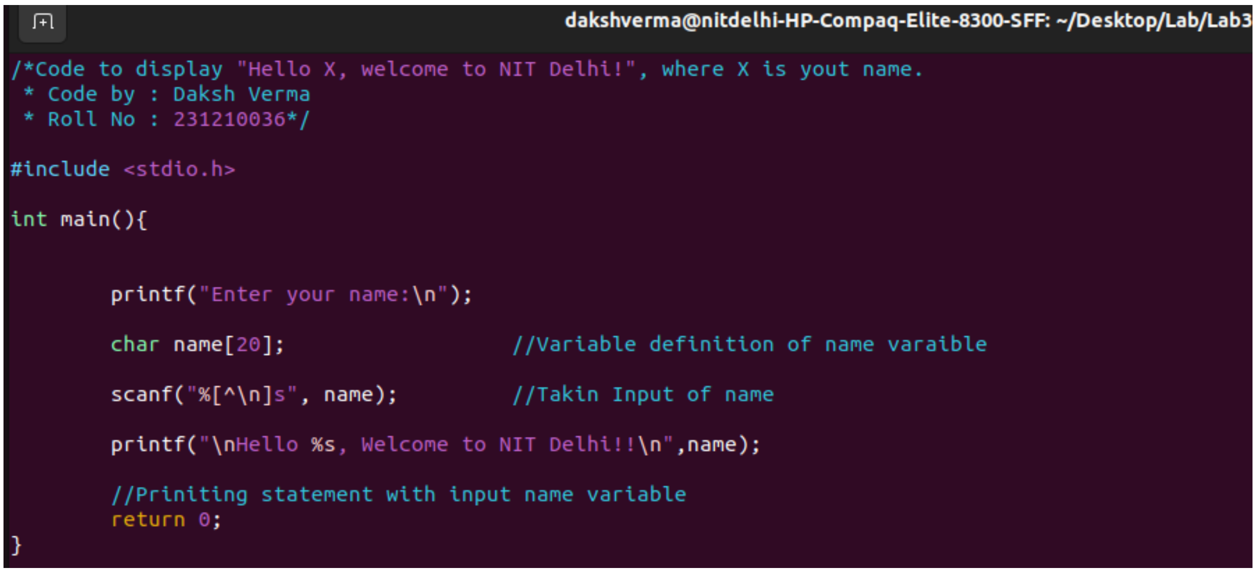
The problem is to write a C program that takes the user's name as input and then prints a welcome message in the format "Hello X, Welcome to NIT Delhi!!," where X is the user's name.

Input Variable	**Processing variable/ Calculations**	**Output variables**	**Necessary header files/functions/macros**
Name (char/string)	none	None	Stdio.h printf()

Algorithm:

1. Start
2. Declare a character array **name** to store the user's name.
3. Prompt the user to input their name.
4. Read and store the user's name in the **name** array.
5. Display the welcome message using **printf()** with the user's name.
6. Stop

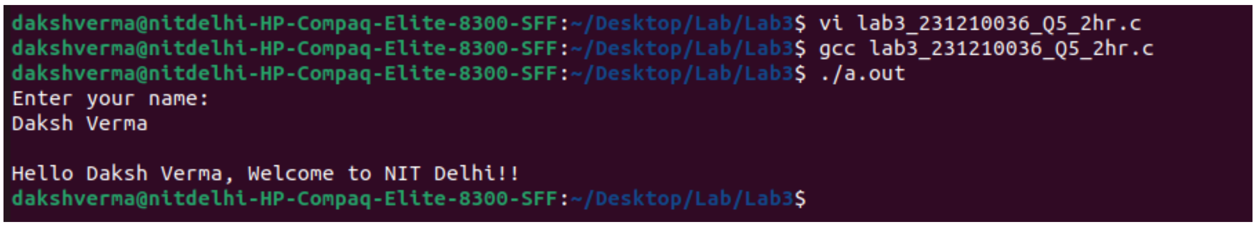
Code



```
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF: ~/Desktop/Lab/Lab3
/*Code to display "Hello X, welcome to NIT Delhi!", where X is yout name.
 * Code by : Daksh Verma
 * Roll No : 231210036*/

#include <stdio.h>

int main(){


        printf("Enter your name:\n");

        char name[20];                  //Variable definition of name varaible

        scanf("%[^\n]s", name);         //Takin Input of name

        printf("\nHello %s, Welcome to NIT Delhi!!\n",name);

        //Priniting statement with input name variable
        return 0;
}
```

Daksh Verma
231210036

Output



```
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ vi lab3_231210036_Q5_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ gcc lab3_231210036_Q5_2hr.c
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$ ./a.out
Enter your name:
Daksh Verma

Hello Daksh Verma, Welcome to NIT Delhi!!
dakshverma@nitdelhi-HP-Compaq-Elite-8300-SFF:~/Desktop/Lab/Lab3$
```

Discussion and Conclusion:

This C program demonstrates user input, character arrays, and formatted output using **printf()**. It provides a simple way to interact with users and personalize messages.

Daksh Verma
231210036

6. Write a program that takes your Roll no and First name as input and print in Decimal,Octal ad Hex using printf() in C.

[**Hint** : Here, following format specifies are used:
%d - to print value in integer format
%o - to print value in octal format
%x - to print value in hexadecimal format (letters will print in lowercase)
%X - to print value in hexadecimal format (letters will print in uppercase)]

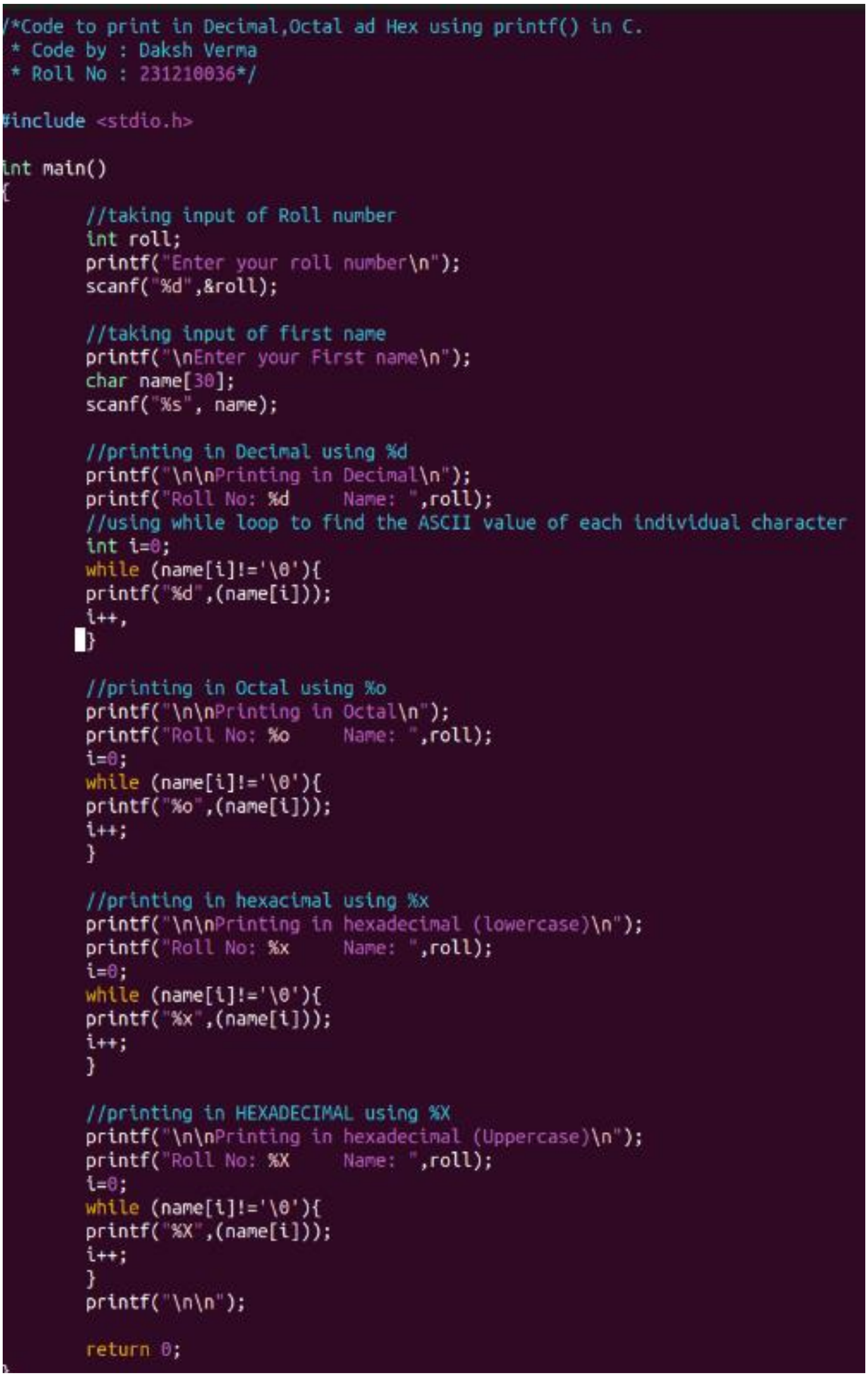
Problem Analysis:
The problem is to write a C program that takes the user's roll number and first name as input and prints these values in decimal, octal, and hexadecimal formats using **printf()**.

Input Variable	**Processing variable/Calculations**	**Output variables**	**Necessary header files/functions/macros**
Roll (int), Name (char)	none	none	Stdio.h printf()

Algorithm:

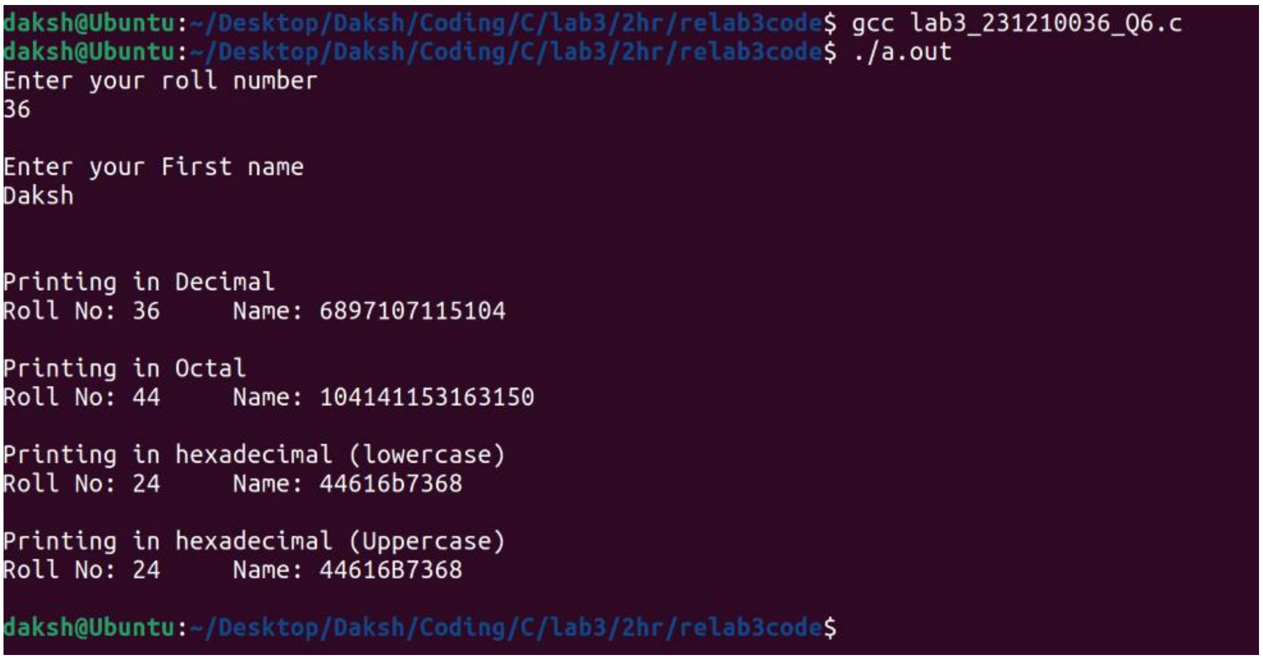
1. Start
2. Declare integer variable **roll**.
3. Declare a character array **name** to store the user's first name.
4. Prompt the user to input their roll number and first name.
5. Read and store the user's roll number in **roll**.
6. Read and store the user's first name in **name**.
7. Display the roll number and first name in decimal, octal, and hexadecimal formats using **printf()**.
8. Run a while loop for string name till '\0' and individually print ASCII value of each character in decimal, octal and hexadecimal
9. Stop

Code



```
/*Code to print in Decimal,Octal ad Hex using printf() in C.
 * Code by : Daksh Verma
 * Roll No : 231210036*/

#include <stdio.h>

int main()
{
        //taking input of Roll number
        int roll;
        printf("Enter your roll number\n");
        scanf("%d",&roll);

        //taking input of first name
        printf("\nEnter your First name\n");
        char name[30];
        scanf("%s", name);

        //printing in Decimal using %d
        printf("\n\nPrinting in Decimal\n");
        printf("Roll No: %d     Name: ",roll);
        //using while loop to find the ASCII value of each individual character
        int i=0;
        while (name[i]!='\0'){
        printf("%d",(name[i]));
        i++,
        }

        //printing in Octal using %o
        printf("\n\nPrinting in Octal\n");
        printf("Roll No: %o     Name: ",roll);
        i=0;
        while (name[i]!='\0'){
        printf("%o",(name[i]));
        i++;
        }

        //printing in hexacimal using %x
        printf("\n\nPrinting in hexadecimal (lowercase)\n");
        printf("Roll No: %x     Name: ",roll);
        i=0;
        while (name[i]!='\0'){
        printf("%x",(name[i]));
        i++;
        }

        //printing in HEXADECIMAL using %X
        printf("\n\nPrinting in hexadecimal (Uppercase)\n");
        printf("Roll No: %X     Name: ",roll);
        i=0;
        while (name[i]!='\0'){
        printf("%X",(name[i]));
        i++;
        }
        printf("\n\n");

        return 0;
}
```

Output



```
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$ gcc lab3_231210036_Q6.c
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$ ./a.out
Enter your roll number
36

Enter your First name
Daksh


Printing in Decimal
Roll No: 36     Name: 6897107115104

Printing in Octal
Roll No: 44     Name: 104141153163150

Printing in hexadecimal (lowercase)
Roll No: 24     Name: 44616b7368

Printing in hexadecimal (Uppercase)
Roll No: 24     Name: 44616B7368

daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$
```

Discussion and Conclusion:
This C program demonstrates user input, character arrays, and formatted output using **printf()** with different format specifiers (%d, %o, %x). It also shows how to calculate the length of a character array. The program allows users to see their input in multiple numeric formats.

Daksh Verma
231210036

7. Write a program that shows the difference between float and double data type.

[***Hint***: *You may use float and double data types to calculate the roots of thisquadratic equation x2 – 4.0000000 x + 3.9999999 = 0*]

Input Variable	**Processing variable/**	**Output variables**	**Necessary header files/functions/macros**
none	none	A,b,c (float) Ddisc,fdisc,sddiscriminant, fsdiscriminant,root1f,root2f,root1d, root2d	Stdio.h printf()

Problem Analysis:

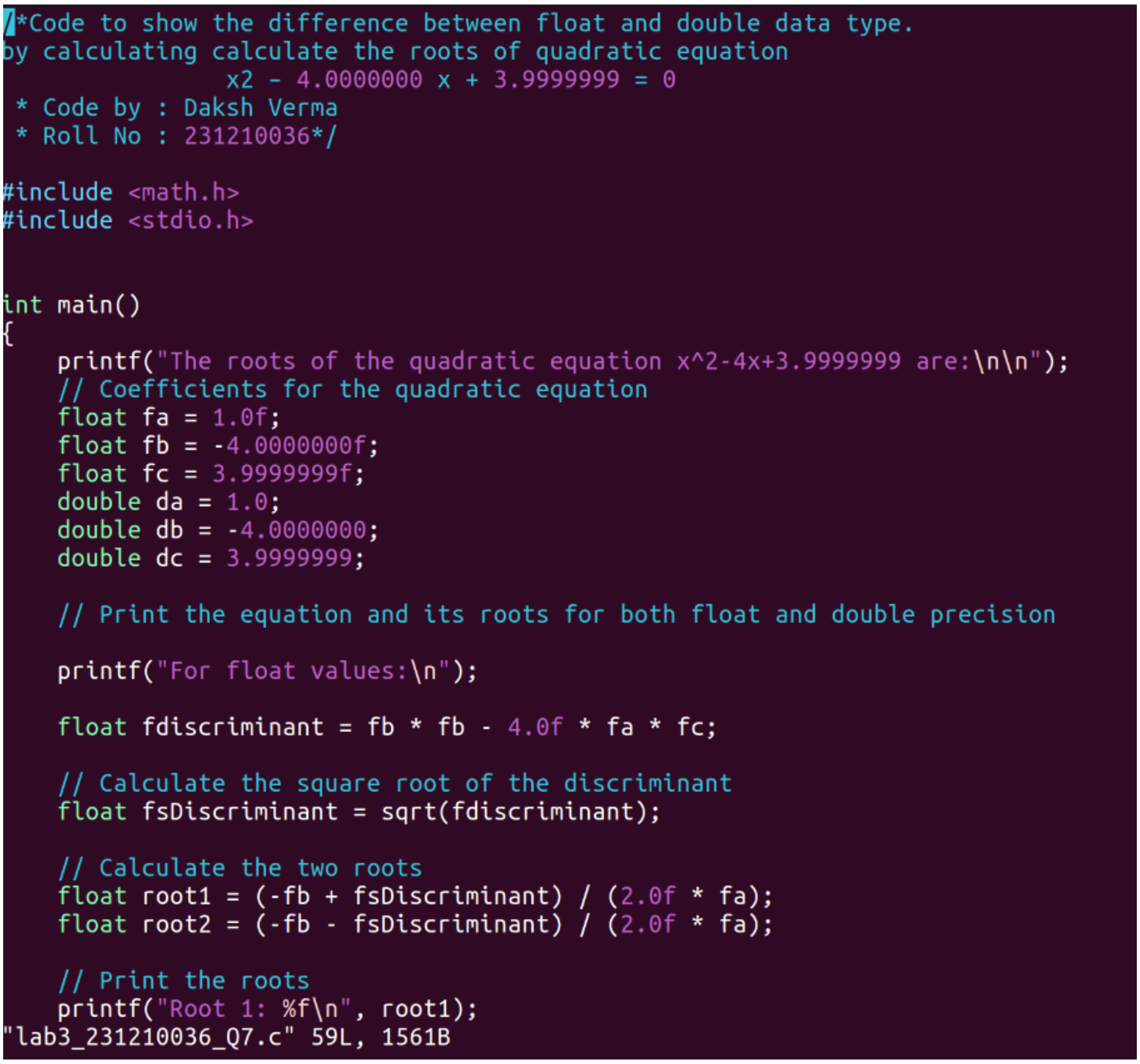
The problem is to write a C program that calculates and compares the roots of a quadratic equation using the float and double data types.

Algorithm:

1. Start
2. Declare floating-point variables a, b, and c for the coefficients of the quadratic equation.
3. Declare variables discriminant_float and discriminant_double for storing the discriminants using float and double data types.
4. Declare variables root1_float, root2_float, root1_double, and root2double to store the roots.
5. Prompt the user to input the coefficients a, b, and c.
6. Calculate the discriminant for both float and double data types.
7. Calculate the roots for both float and double data types.
8. Display the roots using printf() for both data types.
9. Calculate and display the absolute difference between the roots for both data types.
10. Stop

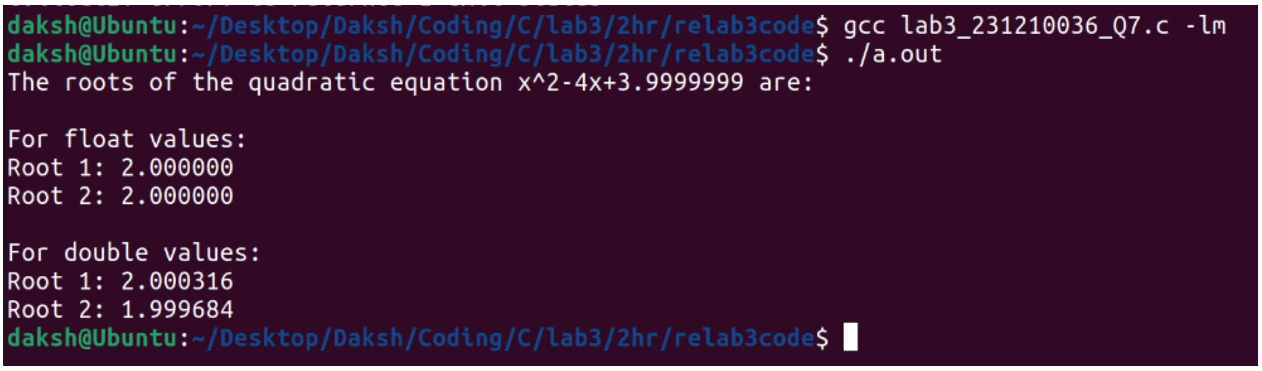
Daksh Verma
231210036

Code



```
/*Code to show the difference between float and double data type.
by calculating calculate the roots of quadratic equation
                x2 - 4.0000000 x + 3.9999999 = 0
 * Code by : Daksh Verma
 * Roll No : 231210036*/

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


int main()
{
    printf("The roots of the quadratic equation x^2-4x+3.9999999 are:\n\n");
    // Coefficients for the quadratic equation
    float fa = 1.0f;
    float fb = -4.0000000f;
    float fc = 3.9999999f;
    double da = 1.0;
    double db = -4.0000000;
    double dc = 3.9999999;

    // Print the equation and its roots for both float and double precision

    printf("For float values:\n");

    float fdiscriminant = fb * fb - 4.0f * fa * fc;

    // Calculate the square root of the discriminant
    float fsDiscriminant = sqrt(fdiscriminant);

    // Calculate the two roots
    float root1 = (-fb + fsDiscriminant) / (2.0f * fa);
    float root2 = (-fb - fsDiscriminant) / (2.0f * fa);

    // Print the roots
    printf("Root 1: %f\n", root1);
"lab3_231210036_Q7.c" 59L, 1561B
```

Output



```
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$ gcc lab3_231210036_Q7.c -lm
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$ ./a.out
The roots of the quadratic equation x^2-4x+3.9999999 are:

For float values:
Root 1: 2.000000
Root 2: 2.000000

For double values:
Root 1: 2.000316
Root 2: 1.999684
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$
```

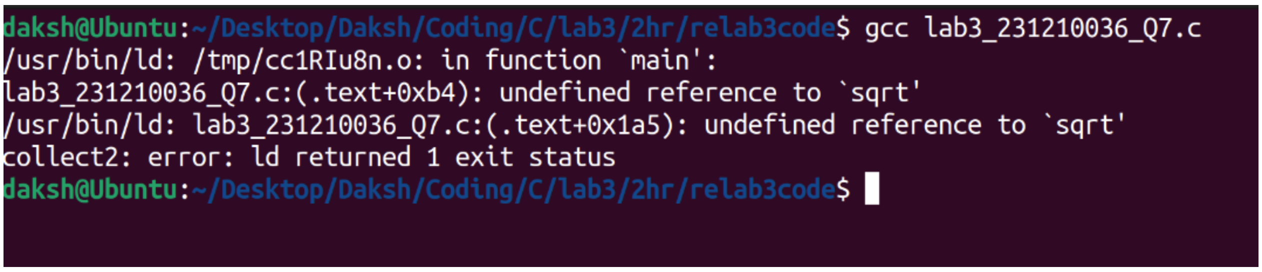
Discussion and Conclusion:

This C program calculates the roots of a quadratic equation using both float and double data types. It demonstrates the precision difference between these data types when dealing with real numbers. The absolute difference between the roots shows how the choice of data type affects accuracy in calculations.

Observation /Comments:

These C programming problems provide a comprehensive exploration of fundamental concepts. From variable manipulation and personalized user interactions to formatted output and data type precision, they offer valuable insights into essential programming skills.

#error



```
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$ gcc lab3_231210036_Q7.c
/usr/bin/ld: /tmp/cc1RIu8n.o: in function `main':
lab3_231210036_Q7.c:(.text+0xb4): undefined reference to `sqrt'
/usr/bin/ld: lab3_231210036_Q7.c:(.text+0x1a5): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
daksh@Ubuntu:~/Desktop/Daksh/Coding/C/lab3/2hr/relab3code$ 
```

I had trouble using the sqrt function in the math header file. The compiler command gcc wasn’t compiling he file. Later I realized that I need to add “—lm” to link the math header file and using “gcc <filename> -lm”, I was able to compile and run the file successfully.