Year: B.Tech II, Computer Organization Lab

Batch (A1)

Practical: **02**

Problem Statements

1. To declare the native data type, check the size of them and print the values if given beyond the range.

Code:

```c
//To declare the native data type,
// check the size of them and
// print the values if given beyond the range.
#include <stdio.h>

int main()
{
    long long int x;
    printf("Enter Integer :\n");
    scanf("%lld", &x);
    if (x >= -2147483648 && x <= 2147483647)
    {
        printf("Valid Input!\n");
        printf("The Size of Integer is %d\n", sizeof(int));
    }
    else
    {
        printf("Invalid Input!\n");
        printf("Value Entered is Out of Range of Data Type\n");
    }

    return 0;
}
```

Sample Test Cases:

1.) [-2147483648]

```
Enter Integer :
-2147483648
Valid Input!
The Size of Integer is 4
```

2.) [2147483647]

```
Enter Integer :
2147483647
Valid Input!
The Size of Integer is 4
```

3.) [2147483648]

```
Enter Integer :
2147483648
Invalid Input!
Value Entered is Out of Range of Data Type
```

4.) [-9999999999]

```
Enter Integer :
-9999999999
Invalid Input!
Value Entered is Out of Range of Data Type
```

2. To perform all arithmetic operations of two numbers given from the command line.

Code:

```c
//To perform all arithmetic operations of two numbers
// given from the command line.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int a, b;

    if (argc < 3)
    {
        printf("Enter Two Arguments!\n");
        return -1;
    }

    a = atoi(argv[1]);

    b = atoi(argv[2]);

    int add = a + b;
    printf("a + b = %d\n", add);
```

```
    int sub = a - b;
    printf("a - b = %d\n", sub);
    int mul = (a * b);
    printf("a * b = %d\n", mul);

    if (b != 0)
    {
        int ans = (a / b);
        printf("a / b = %d\n", ans);
    }
    else
    {
        printf("Divide by Zero Error!\n");
    }

    if (b != 0)
    {
        int ans1 = a % b;
        printf("a %% b = %d", ans1);
    }
    else
    {
        printf("Modulo by Zero Error!");
    }

    return 0;
}
```

Sample Test Cases:

1.) [135, 5]

```
C:\Users\Admin\Desktop\Lab_2>Q2 135 5
a + b = 140
a - b = 130
a * b = 675
a / b = 27
a % b = 0
```

2.) [150, 0]

```
C:\Users\Admin\Desktop\Lab_2>Q2 150 0
a + b = 150
a - b = 150
a * b = 0
Divide by Zero Error!
Modulo by Zero Error!
```

3.) Insufficient Inputs from User

```
C:\Users\Admin\Desktop\Lab_2>Q2 190
Enter Two Arguments!
```

3. To perform all arithmetic operations of two numbers given from the command line, but using qualifiers. Using Qualifier “const”.

Code:

```c
//To perform all arithmetic operations of two numbers
// given from the command line, but using qualifiers.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    if (argc > 1)
    {
        printf("Constants Value Cant Be Modified!\n");
        printf("Enter 0 Arguments Only!");
        return -1;
    }
    // Using Qualifier
    const double a = 250.800, b = 125.400;
    printf("a = %lf\n", a);
    printf("b = %lf\n", b);

    const double add = a + b;
    printf("a + b = %lf\n", add);
    const double sub = a - b;
    printf("a - b = %lf\n", sub);
    const double mul = (a * b);
    printf("a * b = %lf\n", mul);

    if (b != 0)
    {
        const double ans = (a / b);
        printf("a / b = %lf\n", ans);
    }
    else
    {
        printf("Divide by Zero Error!\n");
    }

    printf("Since Modulus Operator Works only for Integer Numbers\nEg : [1000, 3]\n");
```

```
    const long long int num1 = 1000, num2 = 3;

    if (num2 != 0)
    {
        const long long int ans1 = num1 % num2;
        printf("num1 %% num2 = %lld", ans1);
    }
    else
    {
        printf("Modulo by Zero Error!");
    }
    return 0;
}
```

Sample Test Cases:

1.) [] ~ No Inputs Required

```
C:\Users\Admin\Desktop\Lab_2>Q3
a = 250.800000
b = 125.400000
a + b = 376.200000
a - b = 125.400000
a * b = 31450.320000
a / b = 2.000000
Since Modulus Operator Works only for Integer Numbers
Eg : [1000, 3]
num1 % num2 = 1
```

2.) Constants Value can’t be Modified Error in Case User Inputs Data

```
C:\Users\Admin\Desktop\Lab_2>Q3 12
Constants Value Cant Be Modified!
Enter 0 Arguments Only!
```

4. To display different formatting of floating point numbers.

Code:

```
//To display different formatting of floating point numbers.
#include<stdio.h>

int main()
{
    float num;
    printf("Enter a Number(Float Data Type) : \n");
    scanf("%f", &num);
```

```c
    // 1 digits
    printf("1 Digit Float Form : %0.1f\n", num);

    // 2 digits
    printf("2 Digit Float Form : %0.2f\n", num);

    // 3 digits
    printf("3 Digit Float Form : %0.3f\n", num);

    // 4 digits
    printf("4 Digit Float Form : %0.4f\n", num);

    // 5 digits
    printf("5 Digit Float Form : %0.5f\n", num);

    // Exponential Form
    printf("Exponential Form : %e\n", num);


    return 0;
}
```

Sample Test Cases:

```
Enter a Number(Float Data Type) :
23345.9358934
1 Digit Float Form : 23345.9
2 Digit Float Form : 23345.94
3 Digit Float Form : 23345.936
4 Digit Float Form : 23345.9355
5 Digit Float Form : 23345.93555
Exponential Form : 2.334594e+004
```

5. Perform Addition and Subtraction of two signed binary numbers given from command line.

Code:

```c
//Perform Addition and Subtraction of two signed binary numbers given from command line.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int fromBinary(const char *s)
{
    return (int)strtol(s, NULL, 2);
}

int main(int argc, char *argv[])
{
```

```
    int num1, num2, bin_add, bin_sub;

    if (argc < 3)
    {
        printf("Enter Two Arguments!\n");
        return -1;
    }

    num1 = fromBinary(argv[1]);

    num2 = fromBinary(argv[2]);

    bin_add = num1 + num2;
    bin_sub = num1 - num2;

    printf("Binary Addition Result : %d\n", bin_add);
    printf("Binary Subtraction Result : %d\n", bin_sub);

    return 0;
}
```

Sample Test Cases:

1.) [10 (1010), 15(1111)]

```
C:\Users\Admin\Desktop\Lab_2>Q5 1010 1111
Binary Addition Result : 25
Binary Subtraction Result : -5
```

2.) [42 (101010), 14 (1110)]

```
C:\Users\Admin\Desktop\Lab_2>Q5 101010 1110
Binary Addition Result : 56
Binary Subtraction Result : 28
```

6. Perform Multiplication for unsigned binary number.

Code:

```
//Perform Multiplication for unsigned binary number
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```c
int fromBinary(const char *s)
{
    return (int)strtol(s, NULL, 2);
}

int main()
{

    char num1[32], num2[32];
    int n1, n2;

    printf("Input Number 1: ");
    scanf("%s", num1);

    n1 = fromBinary(num1);

    printf("Input Number 2: ");
    scanf("%s", num2);

    n2 = fromBinary(num2);

    printf("Multiplcation Result : %d\n", (n1 * n2));

    return 0;
}
```

Sample Test Cases:

1.) [10(1010), 11(1011)]

```
Input Number 1: 1010
Input Number 2: 1011
Multiplcation Result : 110
```

2.) [62(111110), 42(101010)]

```
Input Number 1: 111110
Input Number 2: 101010
Multiplcation Result : 2604
```

Submitted By:

Roll Number: U19CS012 (D-12)

Name: Bhagya Rana