**Assignment # 1**

1. Two numbers are input through the keyboard into two locations C and D. Write a program to interchange the contents of C and D.
2. If a five-digit number is input through the keyboard
   1. Write a program to calculate the sum of its digits. (Hint: Use the modulus operator ‘%’)
   2. Write a program to reverse the number.
   3. Write a program to print a new number by adding one to each of its digits. For example if the number that is input is 12391 then the output should be displayed as 23402.
3. For any character is entered through the keyboard, write a program to determine whether the character entered is a capital letter, a small case letter, a digit or a special symbol. The following table shows the range of ASCII values for various characters.

|  |  |
| --- | --- |
| **Characters** | **ASCII Values in decimal** |
| **A – Z**  **a – z**  **0 - 9**  **Special symbols** | **65 – 90**  **97 – 122**  **48 – 57**  **0 – 47, 58 – 64, 91 – 96, 123 - 127** |

1. Write a program to display the number (1-10) entered through the keyboard using **switch** statement.
2. Write a general-purpose function to convert any given year into its roman equivalent. The following table shows the roman equivalents of decimal numbers:



Example:

Roman equivalent of 1988 is mdcccclxxxviii

Roman equivalent of 1525 is mdxxv

**Assignment # 2**

1. Write a function **power (a, b)** to calculate the value of **ab.**
2. Write a program to compute the sum of elements in an array.
3. Write a program to display the odd and even numbers in an array.
4. Write a program to perform the multiplication of two 4x4 matrices.
5. Write a program to perform the addition of 2-D arrays.

**Exercise 1 Output**

In the introductory program we include the library stdio.h, which contains the function printf. This function is used to produce output in the form of characters that are printed in the terminal. In its simplest form, the function is called with a string as argument: printf("This text is printed to the screen\n"); That is great, but we want our programs to output more than just the fixed strings that the programmer writes in the program.

To print the contents of variables, we add format specifiers to the string and add the variables we want to print as arguments: int number; char letter; printf("%d is an integer and %c is a character\n", number, letter); Different types of variables have different specifiers, all starting with a percentage sign. Common specifiers are %d for integers, %f for floats, %c for characters and %s for strings. To output a percentage sign, we use %%.

**Write a function that outputs:**

**a) The string: One half is 50%**

**b) two integers and their difference.**

**c) two floats and the result of dividing one with the other**

Write a main function that calls your other functions. The output has to be as follows:

$ ./a.out

One half is 50%

The difference between 10 and 3 is 7

1.000000 / 3.000000 is 0.333333

**Exercise 2 Input**

For input we use the function scanf, also from the library stdio.h. The scanf function takes a format string followed by references to where the input should be stored. Example that reads an integer to a variable:

int number;

scanf("%d", &number);

Notice that the & character in front of the varable name. It means that the variable is passed as reference to scanf. It allows scanf to update the value of the variable. If & is not there, the program would likely crash at that point. When reading a string, the & sign can be omitted:

char my\_variable[100];

scanf("%s", my\_variable);

**Write functions that:**

**a) asks for two integers and outputs them and their sum.**

**b) asks for two floats and outputs their product.**

**c) asks for a word and prints it twice on the same row.**

Write a main function that calls your other functions. The output has to be

$./a.out

Give two integers: 12 5

You entered 12 and 5, their sum is: 17

Give two floats: 3.14 2 You entered 3.140000 and 2.000000, their product is: 6.280000

Give a word: Yey!

Yey! Yey!

[.../exercise]$

**Exercise 3 Conditionals**

If-else statements are used to make a program behave differently depending on the program state or user input. As an example, one can use if-statements to make sure that input is sane before performing an operation

int a;

int b;

...

if(b == 0)

{

printf("Error: Divide by zero!\n");

// Code for error handling. ...

}

else

{

printf("Division evaluates to: %d\n", a/b);

}

Write functions that:

a) ask for an integer and output whether the entered number is zero or not.

b) ask for two floats and outputs the largest of the inputs

c) ask for an integer and, if the number is divisible by two, divides it by two, otherwise multiplies it by three and output the result. Here, the modulo operator % is useful.

d) ask for three integers and output whether any of them are equal. Use only one if-else-statement Write a main function that calls your other functions.

The output has to be as follows:

[.../exercise]$ ./a.out

Give an integer: 12

The number you entered does not equal zero

Give two floats: 13.4 20

20.000000 is the largest

Give an integer: 14

Result is: 7

Give three integers: 1 13 1

Some numbers are equal

Output with different input

$ ./a.out

Give an integer: 0

The number you entered equals zero

Give two floats: 13.2 -150

13.200000 is the largest

Give an integer: 7

Result is: 21

Give three integers: 2 5 13

All are unique

**Exercise 4: Loops**

Loops are used to execute a statement or a block of code multiple times. A loop will continue to execute as long as the loop condition is satisfied. These two example loops will print the numbers 1 to 10 on one line and then 11 to 20 on the next line:

int i,j;

i = 1;

while(i < 11)

{

printf("%d ", i);

i=i+1;

}

printf("\n");

for(j=11;j<=20;j++)

{ printf("%d ", j); }

printf("\n");

Write functions that:

a) print all even numbers between 0 and 40.

b) print all the numbers between 1 and 100, with 10 numbers on each line. Use two for loops. All columns should be aligned.

c) ask for a number than prints the number squared. This repeats until the 0 is entered.

Write a main function that calls your other functions.

The output has to be as follows:

$./a.out

Even numbers between 0 and 40:

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40

Numbers 1 to 100:

1 2 3 4 5 6 7 8 9 10

11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30

31 32 33 34 35 36 37 38 39 40

41 42 43 44 45 46 47 48 49 50

51 52 53 54 55 56 57 58 59 60

61 62 63 64 65 66 67 68 69 70

71 72 73 74 75 76 77 78 79 80

81 82 83 84 85 86 87 88 89 90

91 92 93 94 95 96 97 98 99 100

Give a number: 2

The square of 2 is 4

Give a number: 5

The square of 5 is 25

Give a number: 9

The square of 9 is 81

Give a number: 0

You entered zero

**Exercise 5 Loops II**

a) Write a program that asks for a number. Then the program should print 1 through the given number on separate lines.

b) Encapsulate your code in a while-loop that asks the user if he/she would like to run the program again. Note that when reading a character from the input stream, the newline from the previous input is still buffered and considered as input. To discard the newline, start the scanf string with a space like this: scanf(" %c", &input);.

The output has to be as follows:

$ ./a.out

Give a number: 5

1

2

3

4

5

Run again (y/n)?

y

Give a number: 2

1

2

Run again (y/n)?

n

Exiting...

**Exercise 6 Functions**

Functions are a great way to make code reusable, improve the structure of the code and isolate errors. Write functions that:

a) take two floats as argument and returns the minimum of those.

b) take four floats as argument and returns the minimum. Make use of the function defined in a).

c) are the same as in a) and b), but returns the maximum.

d) take four floats as argument and returns their sum.

Write a main function that asks the user for four floats and then outputs the minimum, maximum, their sum and mean value. Use the functions from a) - d) to implement this. The output has to be as follows:

$./a.out

Give four floats: 10.0 -2.3 13.2 20.4

min: -2.300000

max: 20.400000

sum: 41.299999

mean: 10.325000

**Exercise 7 Functions II**

a) Write functions for the four basic mathematical operations addition, subtraction, multiplication and division. Each function has two numbers as parameters and returns the result. Use integers. You do NOT have to do rounding for the division.

b) Write a program that asks the user for numbers a and b, and then use these numbers as arguments for your functions and print the result on the screen.

The output has to be as follows:

$./a.out

Give a: 11

Give b: 5

11 + 5 = 16

11 - 5 = 6

11 \* 5 = 55

11 / 5 = 2

**Exercise 8 Arrays**

In this exercise we look at some basic operations on arrays. Write a C function that:

a) counts the number of 0’s in an integer array. The number of 0’s is returned by the function.

b) prints an array of integers. The integers are printed on one line, enclosed in curly brackets and separated by commas.

c) triples the value of all elements in an array of integers. All functions take two parameters, a pointer to the array of integers and the number of elements in the array.

Write a main function that asks the user to input 10 integers and stores them in an array. Use your other functions to print the initial array, the number of zero-valued elements in the array and the contents of the array when all elements have been tripled. The output has to be as follows:

$./a.out

Input 10 numbers: 1 2 3 0 -3 -2 -1 0 10 11

Initial array: { 1, 2, 3, 0, -3, -2, -1, 0, 10, 11 }

Number of 0’s: 2

Tripled array: { 3, 6, 9, 0, -9, -6, -3, 0, 30, 33 }

**Exercise 12 Recursion**

The Fibonacci sequence is a sequence of numbers where the first two numbers are 1 and 1 and the next number in the sequence is the sum of the

previous two numbers. The n’th number in the sequence can be calculated as:

f(1) = 1

f(2) = 1

f(n) = f(n − 1) + f(n − 2) See the example for the seven first numbers in the sequence

a) Write a C function with a parameter n that returns the n’th Fibonacci number. The function must be recursive, i.e., it should call itself.

b) Write a program that asks the user for a number n and then prints the n first numbers in the Fibonacci sequence.

The output has to be as follows:

$./a.out

Give n: 7

1

1

2

3

5

8

13