HIGH PERFORMANCE PROGRAMMING
UPPSALA UNIVERSITY
SPRING 2021
LAB 2: PROGRAMMING IN C

The aim of this lab is to repeat and practice the fundamental concepts of the programming language C. In this lab you will write your own code for each given task, no external files from the Studentportalen required for solving tasks are provided. Create a separate directory for each task and write the code for each task in the corresponding directory, for example

```
mkdir Task-1
cd Task-1
```

Consult slides of Lecture 2 provided on Studentportalen and search for information on the web.

Note. You are not required to write makefiles in order to compile your code.

1. C basics

Task 1: Write a C program which prints numbers 100, 96, 92, ..., 0 using

- `for` loop
- `while` loop
- `do...while` loop

Task 2: Write a C program which reads two integer numbers a and b from the standard input using `scanf` function. Output using `printf` function and symbols '.' and '*' the rectangle of size $a \times b$. Example:

```
Input: 5 7
Output:
*******
*.....*
*.....*
*.....*
*******
```

Task 3:

- Write a C program which reads two integers from the standard input. If both numbers are even, write their sum, otherwise, output their product.
- Write a C program which reads three real numbers from the standard input. Find which of these numbers is the largest by absolute value and print the result.

Date: December 17, 2020.

- Write a C program which reads an integer value from the standard input and checks if the first digit is odd or even.

Task 4:

Write a simple calculator. The program can read from the standard input the following command: `number OP number`, where number is a real number and `OP` is a symbol '+', '-', '*' or '/'. Your program must interpret the input and write the corresponding output of the operation.

Example:

```
Input: 3+5
Output: 8
```

Note. Use a `switch` statement for choosing the operation, and write an error message if the value of `OP` is not valid.

Task 5:

A perfect square is an integer that is the square of an integer, for example the numbers 4 and 25 are perfect squares since $4 = 2^2$ and $25 = 5^2$. Write a C program which is checking if the number entered by the user is a perfect square. You can use the `sqrt` function from header `<math.h>`.

Note. The math functions in `<math.h>` have implementations in the library `libm.so`. If your program includes `<math.h>`, then you need to explicitly link to the math library:

```
gcc -o prog prog.c -lm
```

where `prog.c` is a name of your source file.

Task 6:

Write a C program to compute quotient and remainder of two integer numbers entered by the user. Example:

```
Enter dividend: 25
Enter divisor: 4
Quotient = 6
Remainder = 1
```

Hint. Use operators `%` and `/`.

Task 7:

An integer is a palindrome if the reverse of that number is equal to the original number. Check if the number entered by the user is a palindrome.

Example:

```
Input: 12345
Output: it is not a palindrome
Input: 46364
Output: it is a palindrome
```

Hint. Use operators `%` and `/`.

2. Pointers and memory allocation

Task 8:

Write a short C program that declares and initializes (to any value you like) a double, an int, and a char. Declare and initialize a pointer to each of the three variables. Output the value of each input variable, its address in hexadecimal format, and its memory size (in bytes).

Task 9:

The `main` function in a C program is the following:

```
int main()
{
int a,b;
char *s1,*s2;

a = 3; b=4;
swap_nums(&a,&b);
printf("a=%d, b=%d\n", a, b);

s1 = "second"; s2 = "first";
swap_pointers(&s1,&s2);
printf("s1=%s, s2=%s\n", s1, s2);
return 0;
}
```

The function `swap_nums` swaps values of two integers, and the function `swap_pointers` swaps values of two pointers. Write functions `swap_nums` and `swap_pointers` such that the program works and gives the following output:

```
a=4, b=3
s1=first, s2=second
```

Task 10:

The `main` function in a C program is the following:

```
int main()
{
int *arr;
int n;
scanf("%d", &n);
arr = (int *)malloc(n*sizeof(int));
for(int i=0; i<n; ++i) arr[i] = rand() % 100; // random number from 0 to 99
print_array(arr, n);
return 0;
}
```

Write a function `print_array` which prints out all values of an integer array `arr`.

Task 11:

Write a C program which reads numbers from the standard input into an array until something which is not an integer number appears. Then print all entered numbers and compute their sum. Note, the amount of numbers is not known, i.e. the length of the array is not known beforehand. The program should work for any number of input parameters, no maximum array size should be assumed.

Example:

```
Input: 4 5 6 7 34 5 y 3 4 3 6
Output:
4 5 6 7 34 5
Sum: 61
```

Hint. Use `malloc` and `realloc` from the header `<stdlib.h>`.

Task 12:

Write a C program which reads a number n from the standard input and then reads n integer numbers into an array. Remove all prime numbers from the array putting the results in a new array. Output the elements of the new array and its size. The program should work for any number n, no maximum array size should be assumed.

Hint. Use `malloc` and `realloc` from the header `<stdlib.h>`.

3. File operations

Part 1.

Create a file `data.txt` with the following content:

```
5
Milk 10.3
Water 5.2
Potatoes 3.1
Carrots 4.8
Meat 20.0
```

The number written in the first line of the file is a total number of products. All following lines contain a name of a product and the corresponding price. The name of each product is a string of length not exceeding 50 characters.

Task 13:

Write a C program which reads data from a file "data.txt" line by line and directly outputs it to the standard output as a table:

```
Milk 10.3
Water 5.2
Potatoes 3.1
Carrots 4.8
Meat 20.0
```

Task 14:

Modify your program such that is accepts one argument passed from the command line. This parameter will be a file name with a list of products. Make sure your program can be run like this:

```
./a.out  data.txt
```

Write an error message if the requested file does not exist.

Task 15:

Let product be represented in a code as a structure:

```
typedef struct product
{
char   name[50];
double price;
}
product_t;
```

Modify your program such that all data read from a file are stored into an array of products:

```
product_t arr_of_prod[100];
```

Assume there will be no more that 100 products in a file. When all data are read from the file, output data from the array to the standard output as a table.

Task 16:

Assume that the number of products in a file is unknown and can be very big. Modify your program such that the array `product_t *arr_of_prod` is allocated dynamically using `malloc` as soon as you know the exact number of products in the file.

Part 2.

Task 17:

Write a program that copies one file to another, converting lower case letters to upper case. Your program should accept both filenames as arguments passed from the command line. Use functions `fgetc`, `fputc` from the header `<stdio.h>` and `toupper` from the header `<ctype.h>`. For example if the input file is:

in.txt:

```
hello world
```

Then running

```
./a.out  in.txt out.txt
```

we get in out.txt:

```
HELLO WORLD
```

4. Extra part

Task 18:

(Recursive function) Each number can be represented as a sum of non-descreasing squares. For example for n = 9 we can have 4 such sums (including the number 9 itself): 9, 1 + 4 + 4, 1 + 1 + 1 + 1 + 1 + 4, 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1.

Write a C program which is counting such sums for a given number.

Examples:

```
Enter number: 9
Number of sums: 4

Enter number: 16
Number of sums: 8

Enter number: 123
Number of sums: 2641

Enter number: 326
Number of sums: 481771
```