University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Informatik II
Assignment 1

Feb 19, 2018

Note: Task 1 and Task 6 have to be completed in the first lab with the guidance of the tutors.

Introduction to C [15 points]

Task 1. Given a string of n characters and a string of m characters, write a program in C that uses the function **int isSubstring(char str1[], char str2[])** to *check* if `str1` is a *substring* of `str2`. Your program should prompt the user to type 2 strings and then print the result. An input/output example is illustrated below (input is typeset in bold):

```
Enter the first string: thesis
Enter the second string: hypothesis
Is Substring: True
```

Task 2 [5 points]. Consider an array `A[0...n-1]` with `n` natural values. Write a program in C that prompts the user to type the values of A, it computes the sums of even and odd numbers respectively, and it prints which set has the biggest sum. An input/output example is illustrated below (input is typeset in bold):

```
Values of A separated by spaces (non-number to stop): 1 2 3 4 5 end
Even sum : 6
Odd sum : 9
Largest sum : odd
```

Task 3 [10 points]. Consider an array `A[0...n-1]` with `n` natural values. Write a program in C that reads array `A` and then prints the prime numbers contained in `A` using the function **int isPrime(int a)**. Function **isPrime** must get an integer `a`, test whether `a` is multiple of any integer between 2 and $\sqrt{a}$ and return 0 or 1 respectively. An input/output example is illustrated below (input is typeset in bold):

```
Values of A separated by spaces (non-number to stop): 7 20 37 49 53 end
Prime numbers : 7 37 53
```

Basic Sorting [20 points]

Task 4 [10 points]. Write a program in C that reads a string and uses bubble sort to sort its characters in alphabetical order (uppercase characters shall precede the corresponding lowercase characters).

(a) Implement the function **int compare(char a, char b)** that compares characters in this manner, and a function **void bubblesort(char a[], int n)** that uses `compare` to sort the characters accordingly. Function **int compare(char a, char b)** returns -1 if `a` precedes `b` in the sorted result, 1 if a succeeds b, and 0 otherwise. An input/output example is illustrated below (input is typeset in bold):

```
Type a string: AlgoDat
Sorted string: AaDglot
```

(b) Implement the function **int isAnagram(char a[], char b[])** to *check* if the first string is an *anagram* of the second (case-sensitive). Your program should prompt the user to type 2 strings and then print the result. An input/output example is illustrated below (input is typeset in bold):

```
Enter the first string: listen
Enter the second string: silent
Is Anagram: True
```

Task 5 [10 points]. Consider an array `A[0...n-1]` with `n` integer values. Write a program in C that reads array `A` and then sorts its elements using the function **void efficientSort(int A[], int n)**. Efficient Sort is an improved Bubble Sort. Efficient Sort is similar to Bubble Sort but additionally swaps elements while traversing the array from the end to the beginning as well. An input/output example is illustrated below (input is typeset in bold):

```
Values of A separated by spaces (non-number to stop): 5 4 3 2 1 end
Sorted Array : 1 2 3 4 5
```

Recursion [25 points]

Task 6. Given positive integers b and e, write a C program that uses the function **int power(int base, int exponent)** to calculate b^e recursively. An input/output example is illustrated below (input is typeset in bold):

```
Enter the base: 2
Enter the exponent: 5
Result: 32
```

Task 7 [8 points]. Given a positive integer value n, write a program that uses the function **void sequence(int n)** to compute and print all elements of the following sequence. For a given integer n the next integer n' of the sequence is computed as follows:

- If n is even, then $n' = \frac{n}{2}$
- If n is odd, then $n' = 3n + 1$

The sequence stops if $n = 1$. An input/output example is illustrated below (input is typeset in bold):

```
Enter the first sequence number: 3
Sequence : 3 10 5 16 8 4 2 1
```

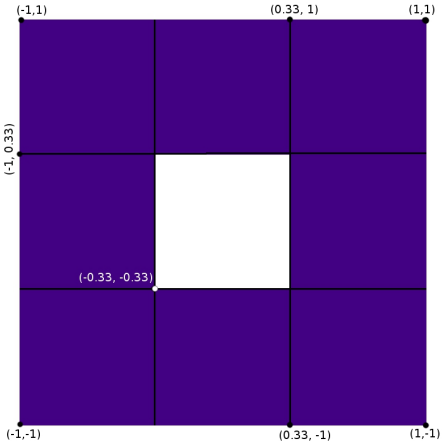
Task 8 [17 points]. The Sierpinski Carpet is a plane fractal. Its initial iteration is constructed by dividing a square into nine subsquares, omiting the central subsquare and draw the remaining eight subsquares. Assuming that SC_i is the Sierpinski Carpet of the i-th iteration, SC_i is constructed by dividing it into nine subsquares, omiting the central subsquare and construct SC_{i-1} for the remaining eight subsquares. Figure 1 depicts Sierpinski Carpet of iterations 1, 2 and 3.
A square is defined by three parameters. The coordinates (`x`, `y`) of its lower-left corner and its side length (`l`).

(a) Create the C function **drawSierpinskiCarpet(double x, double y, double l, int iter)**. The function should calculate the coordinates of the lower-left corner and the side length that define each of the nine subsquares, do appropriate recursive calls for each of the eight remaining subsquares if the number of iterations left is greater than zero or print the coordinates and side length of the subsquares that must be drawn.

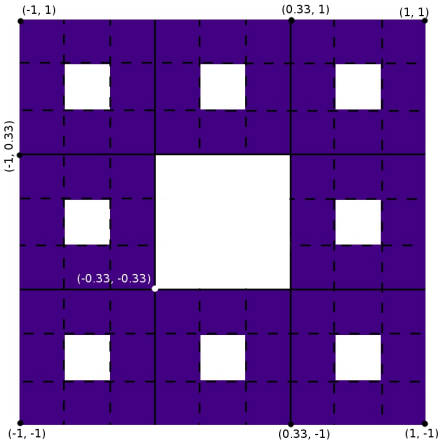
Example: The invocation **drawSierpinskiCarpet(-1.0, -1.0, 2.0, 1)** prints the coordinates of the eight remaining subsquares (after omiting the central one) in order for the Sierpinski Carpet of the first iteration to be drawn. The output would be:

```
(-1.00, -1.00), 0.67
(-0.33, -1.00), 0.67
( 0.33, -1.00), 0.67
(-1.00, -0.33), 0.67
( 0.33, -0.33), 0.67
(-1.00, 0.33), 0.67
(-0.33, 0.33), 0.67
( 0.33, 0.33), 0.67
```

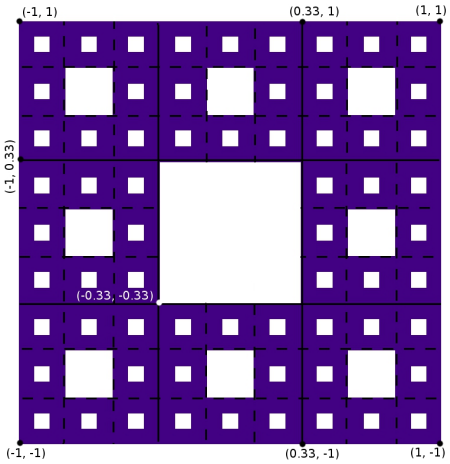
Accordingly, an invocation **drawSierpinskiCarpet(-1.0, -1.0, 2.0, 2)** should print the coordinates of the 64 subsquares needed to draw the carpet presented in Fig. 1(b). [10 points]



(a) Iter=1



(b) Iter=2



(c) Iter=3

Figure 1: Sierpinski Carpet

(b) Write a program in C that adapts the function **drawSierpinskiCarpet(double x, double y, double l, int iter)** to draw the carpet using OpenGL.
[7 points]
Note: A skeleton of the solution of Task 8(b) is provided.

Submission

Please submit a folder *a<exercise number>_<family name>_<matriculation number>.zip* where `family name` and `matriculation number` correspond to your personal data. The folder should include the C files you created for each of the tasks. Each C file should be named *task<task number>.c* and it should include your personal data in the form of a comment on the top.

The zipped folder should be uploaded in the page of the course in OLAT in the ``Drop box'' corresponding to Assignment 1 latest on **Sunday, March 4th at 23:59**.