Programming in C and C++
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH-230-A
November 29th, 2019

Practice Sheet

First Name: .....

Last Name: .....

Matriculation Number: .....

- Read all the following points before proceeding to the solution.
- Write immediately your name on this sheet.
- Write clearly. Take into consideration that C and C++ are case sensitive languages.
- Indent your code in a sensible way.
- Books, slides, notes or other documents are not allowed.
- If you need more space to solve the exercises you may use also the back of each page.
- Read carefully the questions and strictly adhere to the requirements.
- You have two hours to solve this test.
- Any attempt to cheat leads to an immediate fail.
- By signing this sheet you imply you read and understood all of the above.

Signature: .....

%	0.00 - 39.49	39.50 - 44.49	44.50 - 49.49	49.50 - 54.49
Grade	5.0	4.7	4.3	4.0

%	54.50 - 59.49	59.50 - 64.49	64.50 - 69.49	69.50 - 74.49
Grade	3.7	3.3	3.0	2.7

74.50 - 79.49	79.50 - 84.49	84.50 - 89.49	89.50 - 94.49	94.50 - 100.00
2.3	2.0	1.7	1.3	1.0

Reference ASCII Table

Decimal	Character	Decimal	Character	Decimal	Character
32	space	64	@	96	‘
33	!	65	A	97	a
34	”	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	’	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

Reference I/O Functions

```
#include <stdio.h>

#define SEEK_SET    0   /* Seek from beginning of file.  */

#define SEEK_CUR    1   /* Seek from current position.  */

#define SEEK_END    2   /* Seek from end of file.  */

int fflush(FILE *stream);

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);

int fseek(FILE *stream, long offset, int whence);

long ftell(FILE *stream);

int fclose(FILE *stream);

int feof(FILE *stream);

int ferror(FILE *stream);

int fgetc(FILE *stream);

int fgets(const char *s, int size, FILE *stream);

FILE *fopen(const char *path, const char *mode);

int fprintf(FILE *stream, const char *format, ...);

int fputc(int c, FILE *stream);

int fputs(const char *s, FILE *stream);

int fscanf(FILE *stream, const char *format, ...);

int getchar(void);

int printf(const char *format, ...);

int putchar(int c);

int puts(const char *s);

int scanf(const char *format, ...);

int sscanf(const char *str, const char *format, ...);
```

Reference `string.h` Functions

```
#include <string.h>

char *strcat(char *dest, const char *src);

char *strncat(char *dest, const char *src, size_t n);

char *strchr(const char *s, int c);

char *strstr(const char *searchin, const char *what);

int strcmp(const char *s1, const char *s2);

int strncmp(const char *s1, const char *s2, size_t n);

char *strcpy(char *dest, const char *src);

char *strncpy(char *dest, const char *src, size_t n);

size_t strlen(const char *s);

void *memset(void *s, int c, size_t n);

char *strdup(const char *s);

char *strtok(char *str, const char *delim);
```

Reference `stdlib.h` Functions

```
#include <stdlib.h>

void exit(int __status);

void free(void *__ptr);

void *malloc(size_t __size);

void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

Reference Constructors, Methods and Functions (Selection)

```
logic_error(const string& what_arg);

logic_error(const char * what_arg);

exception() throw();

const char * what() const throw();
```

Problem P.1 (2 points)

Language: C

Please complete the following program fragment such that it prints `n` rows with the pattern below using nested loops.

```
int n;
scanf("%d", &n);
// add your code completion below
```

So if the user enters 6 for `n`, the following 6 rows should be printed:

```
A
AB
ABC
ABCD
ABCDE
ABCDEF
```

Problem P.2 (3 points)

Language: C

Write a program which does the following:

a) reads a double from the keyboard,

b) reads a float from the keyboard,

c) reads an integer from the keyboard,

d) stores the product of these three values into the variable `result`,
No information should be lost.

e) prints the value of `result`,

f) uses a pointer `r_ptr` to add 5 to `result`,

g) prints the new values twice, once by using `result`, once by using `r_ptr`.

Problem P.3 (5 points)

Language: C

Write a program where you first read an integer `n`, then `n` integers are read from the keyboard and stored in a dynamically allocated array. Then these numbers and their square should be written to a file named `squares.txt` in the opposite order of their input.
So if the user enters

```
6
1
2
3
4
5
6
```

your output should look like in the following:

```
6 36
5 25
4 16
3 9
2 4
1 1
```

Problem P.4 (3 points)

Language: C

It is suggested that passwords mix letters and digits such that passwords are not that easy to guess. A good password must have at least a minimum length of 8 characters and needs to contain at least three digits.

Please write a function as a password checker which get an array of characters as parameter and determines whether the password is good or not by returning true or false, respectively.

Problem P.5 (2 points)

Language: C

Write the definition of the function

```
bool odd(unsigned char data);
```

which checks is the value passed to the function is odd or not by returning true or false, respectively. The function has to use bitwise operators and is not allowed to use arithmetic operators.

Problem P.6 (2 points)

Language: C

Please write down the output of the following program.

Be exact and write down exactly as it would be printed by a running program.

```
int factorial(int n)
{
    int val;
    if ((n == 0) || (n == 1)) {
        printf("called with par = %d\n", n);
        return 1;
    } else  {
        printf("called with par = %d\n", n);
        val = n * factorial(n - 1);
        printf("returning %d\n", val);
        return val;
    }
}
int main() {
    printf("%d\n", factorial(4));
}
```

Problem P.7 (3+1 = 4 points)

Language: C

Write the definition of the following function

```
int substitute_vowels(char *s, char ch);
```

A vowel is one of the letters `a, e, i, o, u` (it is sufficient to just substitute lowercase characters). The function takes a string and replaces all vowels with the given character `ch`. The function returns the number of replacements done. If a vowel is replaced by the same character, it still counts as a replacement.

You will receive a bonus point if you walk the string using pointers.

Thus the following (incomplete) piece of code

```
char s[] = "This is a sentence";
printf("%s\n", s);
n = substitute_vowels(s, 'o');
printf("%s\n", s)
printf("%d\n", n);
```

will print

```
Thos os o sontonco
6
```

You may **not** use any predefined functions from `string.h` in your program.

Problem P.8 (3 points)

Language: C

We want to record the time a runner completes a lap.
Write the definition of the following function

```
void total_time(int mins[], int secs[], int n, int *sum_min, int *sum_sec);
```

The function accepts two integer vectors containing the lap times in minutes and seconds, and then calculates the sum of all times. The sum expressed in minutes and seconds is copied into two locations pointed by `sum_min` and `sum_sec`.
You can assume that `n` is greater than 1.
The seconds part should not be higher than 59 (thus convert the seconds to minutes if necessary after summing up).

Problem P.9 *A triangle* (3 points)

Language: C

Complete the following program fragment such that the program prints a triangle as described further below.

```
int n;
char ch;
scanf("%d", &n);
getchar();
scanf("%c", &ch);
// add your code completion below
```

Then the program prints a triangle with `n` rows, `n` columns using the character `ch` as illustrated in the testcase.

Testcase: input

```
4
@
```

Testcase: output

```
@
@@
@@@
@@@@
```

Problem P.10 *A matrix* (5 points)

Language: C

Consider (do not write) a program that creates a matrix of `r` rows and `c` columns and then initializes it with values that are read from a file. Then the matrix is printed to the screen.
Write a program fragment which reads the data from a file named `matrix.dat`, which contains the number of rows in the first line, the number of columns in the second line, then values are given for given rows and columns (see input structure and example below).
Also implement and call a function to print the matrix where the prototype looks like this:

```
void print_matrix(int **A, int rows, int cols)
```

Input structure

```
4
3
1 1 3
2 2 5
```

The given input creates are 4 by 3 matrix and sets $1, 1$ to 3 and $2, 2$ to 5.

$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Problem P.11 *A linked list* (6 points)

Language: C

Consider (do not write) a program that reads strings from standard input (the keyboard) and adds them to the beginning of a linked list.
Write a function `struct list*insertBegin(struct node*, char str[])` that implements the insertion of elements into the list.
Also write a function `void printList(strunct node *)` which prints the content of the list with spaces between the elements.
Use the following data structure:

```
struct node {
    char str[20];
    struct list* next;
};
```

Running the program could look like the following:

Testcase: input

```
4
apple
banana
orange
plum
```

Testcase: output

For simplicity **there is a space** after the last string in each row.

```
apple
banana apple
orange banana apple
plum orange banana apple
```

Problem P.12 *Using `qsort()`* (6 points)

Language: C

Consider the following data structure:

```
struct river {
    char name[40];
    int length;
    int drainage_area;
};
```

Write a program fragment that reads data from a file named `data.txt` and then sorts the rivers by length using the predefined `qsort()` function from `stdlib.h`. The result should be written to a file called `output.txt`.
You will also need to implement a comparison function to make `qsort()` work correctly. *You can assume that the `struct` above are part of your program, and that `data.txt` will not contain more than* 30 *entries.*
An example for the content of `data.txt` can be the following:

```
Nile 6650 334900
Amazon 6400 6915000
Yangtze 6300 1800000
Mississippi-Missouri 6275 2980000
Yenisei-Angara-Selenga 5539 2580000
Yellow 5464 745000
Ob-Irtysh 5410 2990000
Congo-Chambeshi 4700 3680000
```

Problem P.13 *A coffee machine* (5 points)

Language: C

A coffee machine is offering its services. Internally the orders by the users are handled via the following `struct`:

```
struct coffee {
    int id;
    char property;
};
```

A coffee can have the following properties:

- The type can either be `regular`, `espresso` or or `double espresso`.
- Milk additives can be `milk`, `cream`, or `soy milk`
- Sweeteners can be `sugar` or `artificial sweetener`Then the order is replaced to have **cream** instead of **milk**.

Therefore, consider the following bit masks:

```
const unsigned char regular = 1<<0;
const unsigned char espresso = 1<<1;
const unsigned char double_espresson = 1<<2;
const unsigned char milk = 1<<3;
const unsigned char cream = 1<<4;
const unsigned char soy_milk = 1<<5;
const unsigned char sugar = 1<<6;
const unsigned char sweetener = 1<<7;
```

To test the coffee machine, please write a program fragment that creates a **regular coffee with milk and sugar**.
Also write the functions

```
void set_coffee_property(struct coffee* c, const unsigned char property)
void unset_coffee_property(struct coffee* c, const unsigned property)
```

to set and unset the properties of a coffee. These functions just need to set/unset the given properties, they do not need to do additional checks. It is the task of the user/machine interface to avoid wrong combinations.

Problem P.14 *Binary Write to File* (3 points)

Write a program fragment which opens a file called `generate.txt` which writes 100 lines into the file using binary write (i.e., `fwrite`) the following data formatted as follows:

```
1 1
2 4
3 9
4 16
...
100 10000
```

Problem P.15 *A Car* (2 points)

Language: C++
A car is characterized by a brand name, model name and a price. Write down the class definition for `Car`. Its properties should not be directly accessible.
Implement inline the parametric constructor for setting all properties, the destructor and the setter for the model name.

Problem P.16 *A taxi* (3 points)

Language: C++
Derive a class `Taxi` from the class `Car` you declared in the previous problem. A Taxi is a car which has a limit on the number of passengers it may carry. Write the declaration and definition of the class, including the parametric constructor and destructor. It should be possible to specify all properties of a Taxi at object creation. The class `Taxi` should also provide a `print` method which prints all properties of a `Taxi` on the screen. Write also an implementation of the `print` method. The class declaration must be consistent with the one you provided while solving the previous problem.

Problem P.17 *Copy Constructor* (3 points)

Language: C++
Consider the declaration of the following classes: (please read carefully the given code):

```
class Book {
    private:
```

```
        const char* title;
        int pages;
    public:
        Book(const Book&);
        Book(const char* t, int p);
        Book();
        // ...
        // setter and getter methods omitted
};
```

a) Write an implementation of the copy constructor for `Book`

b) Normally a copy constructor will be automatically created by the compiler if none is provided. In short words, what does the compiler-created copy constructor do, and why should the programmer provide its own version for the class above?

Problem P.18 *Explanation* (1 point)

Language: C++

Please explain the keywords `private` and `protected` and point out their differences.

Problem P.19 *Overloading operators* (5 points)

Language: C++

Consider the following class (`complex.h`):

```
class Complex {
    private:
        float real;  // real part
        float imag;  // imaginary part
    public:
        Complex();
        Complex(float, float = 0);
        double magnitude();
        void print();
};
```

and the according definition of the print method in this fragment of complex.cpp:

```
void Complex::print() {
    if (imag)
        cout << noshowpos << real << showpos << imag << "i" << endl;
    else
        cout << noshowpos << real << endl;
}
```

Provide an overloaded << operator, which will replace the `print` function and creates the same output.
Also overload the < and > operators so you may compare complex numbers via their magnitude (so just use a `magnitude()` method for comparison (which you do NOT need to implement)).

Add the declarations of the overloaded operators into `complex.h` above. Then just write down the definition below; it will be considered as part of the file `complex.cpp`.

Problem P.20 *A worker class* (4 points)

Language: C++

Write an appropriate class for the program below such that the class declaration, definition and this `main` function will compile and run together.

```
int main() {
  worker a(234, "John McEnroe");
  worker b(324, "Jack Nicholson");

  cout << a << b;
  return 0;
}
```

Problem P.21 *Hierarchy of 3D-objects* (5 points)

Language: C++

Model and write down classes and their relationships (no diagram needed) corresponding to the following entities: `Object3D`, `Sphere`, `Cylinder`, `RectPrism` (i.e., rectangular prism), and `Cube`. An `Object3D` is characterized by a name and the rest are additionally characterized by their mathematical properties.

Make sure that the `Object3D` class is abstract with respect to a method called `volume()` which serves for computing and returning the volume of a 3D-object.

Implement the parametric constructors for `RectPrism` and `Cube` only, and the `volume()` method for all the classes from above such that they will be consistent with the `main()` function from below.

The formulas for computing the volumes of the 3D-objects are:

$$V(Sphere) = \frac{4}{3} \cdot \pi \cdot radius^3$$

$$V(Cylinder) = \pi \cdot radius^2 \cdot height$$

$$V(RectPrism) = width \cdot length \cdot height$$

$$V(Cube) = side^3$$

For π you can use the `M_PI` macro from `cmath`.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

// add your code below

int main(int argc, char *argv[]) {
    Object3D *arr[5];
    arr[0] = new Sphere("sphere", 1.3);
    arr[1] = new RectPrism("rectprism", 2.0, 3.0, 4.0);
    arr[2] = new Cylinder("cylinder", 2.0, 1.3);
    arr[3] = new Cube("cube", 2.0);
    arr[4] = new RectPrism("rectprism2", 1.0, 2.0, 3.5);
    int i;
    for(i=0; i<5; i++)
        cout << arr[i]->volume() << endl;
    for(i=0; i<5; i++)
        delete arr[i];
    return 0;
}
```

Problem P.22 *Multiple inheritance* (3 points)

Language: C++

Consider the program from below. Write down the exact output which this program would produce when running it.

```
#include <iostream>
using namespace std;
class Person {
    protected:
        string name;
        int age;
    public:
        ~Person() {
            cout << "Person destructor" << endl;
```

```
        }
};
class Student : public virtual Person {
    protected:
        long int matr_nr;
    public:
        ~Student() {
            cout << "Student destructor" << endl;
        }
};
class Faculty : public virtual Person {
    protected:
        int employ_nr;
    public:
        ~Faculty() {
            cout << "Faculty destructor" << endl;
        }
};
class TA : public Student, public Faculty {
    private:
        string course_nr;
    public:
        TA() {
            course_nr = "CH_230_A";
        }
        TA(const TA &obj) {
            this->name = obj.name;
            this->age = obj.age;
            this->matr_nr = obj.matr_nr;
            this->employ_nr = obj.employ_nr;
            this -> course_nr = obj.course_nr;
        }
        ~TA() {
            cout << "TA destructor" << endl;
        }
        friend void print(const TA a);
};
void print(const TA a) {
    cout << a.course_nr << endl;
}
int main() {
    TA ta1;
    TA ta2(ta1);
    print(ta2);
    return 0;
}
```

Problem P.23 *Exceptions* (2 points)

Write a program fragment which illustrates the throwing and catching of a `logic_error` while performing a division in the case of division by zero.