Programming in C I
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: JTSK-320111
September 26th, 2018

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 is a case sensitive language.
- 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 - 94.99	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>

int fclose(FILE *stream);

int feof(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 stdlib.h Functions

```
#include <stdlib.h>

void exit(int __status);

void free(void *__ptr);

void *malloc(size_t __size);
```

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);

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);
```

Problem P.1 (3 points)

Please write a program where you first ask for an integer `n`. The program then prints `n` rows with the pattern below using nested loops.
So if you enter 6, 6 rows will be printed:

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

Problem P.2 (3 points)

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 (6 points)

Write a program where you first read an integer n, then n integers are read from the keyboard. Then these numbers and their square should be written to a file named `squares.txt` in the opposite order of their input.
If you do not know how to write to a file, write to standard output instead (you will lose some points then).
So if you enter

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

your output should look like this then:

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

Hints: Dynamically allocate an array. Make sure to cover possible error conditions.

Problem P.4 (5 points)

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

Please write a password checker where you can **repeatedly** enter passwords (they will never be longer than 40 characters). If the string is empty (keep in mind that there might be just a newline character), the program quits and no output is printed.
After each password entered, your program determines whether the password is good or not, and either prints "PASSWORD IS GOOD" or PASSWORD IS BAD" to the screen.

Problem P.5 (3 points)

Write the definition of the function

```
int binaryToDecimal(int b[]);
```

which converts the number `b` from base 2 to base 10. The array `b` consists of 0s and 1s . The number is considered terminated when a digit other than 0 or 1 appears. According to this convention, the following line
`int n[5] = { 1 , 0 , 0 , 1 , 2 };`
defines the binary number 1001, whose value in base 10 is 9 (i.e.,
`binaryToDecimal(n)` should return 9).

Remember that given the binary number $b_n b_{n-1} \ldots b_2 b_1 b_0$, its decimal value is $b_0 \times 2^0 + b_1 \times 2^1 + b_2 \times 2^2 + \ldots b_n \times 2^n$.

Problem P.6 (2 points)

Please write down the output of the following program. (This means: Assume that the program is compiled and executed on a computer, write down the output that the program then generates.)

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)

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 (4 points)

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).