Programming in C II
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: JTSK-320112
February 7th, 2017

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 - 39.99	40.00 - 44.99	45.00 - 49.99	50.00 - 54.99
Grade	5.0	4.7	4.3	4.0

%	55.00 - 59.99	60.00 - 64.99	65.00 - 69.99	70.00 - 74.99
Grade	3.7	3.3	3.0	2.7

75.00 - 79.99	80.00 - 84.99	85.00 - 89.99	90.00 - 94.99	95.00 - 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);

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 *));
```

Problem P.1 *A triangle* (3 points)

Write a program that reads an integer `t` and then the character `ch`. Then the program prints a triangle with `t` rows, `t` columns using the character `ch`.

Testcase: input

```
4
@
```

Testcase: output

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

Problem P.2 *Binary string conversion* (4 points)

Write a function `void itobs(unsigned int no, char *str)`, which receives an unsigned integer and a character pointer as parameters. The function should convert an unsigned integer into its binary string representation, therefore it determines the pattern of 1s and 0s of the unsigned integer and puts this into the string.
Then write a program that repeatedly reads an unsigned integer from the keyboard that is passed to `itobs()`. Then the program prints the resulting string from the `main` function to standard output.

Problem P.3 *A matrix* (5 points)

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.
You will read 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 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.4 *A linked list* (8 points)

Write a program that reads strings from standard input (the keyboard) and adds them to the beginning of a linked list. A *ZZZ* will quit the program.
After each insertion the list should be printed to the screen.
Implement a function `struct list*insertBegin(struct list*, char str[])` that implements the insertion of elements into the list. Also implement a `printList()` function.
Use the following data structure:

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

*You may **not** use global variables in your program.*
(`struct list` itself is a type and not a variable.)

Testcase: input

```
apple
banana
orange
plum
ZZZ
```

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.5 *Using `qsort()`* (7 points)

Consider the following data structure:

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

Write a program that reads data from a file named `data.txt` and then sorts the rivers by length using the predefined `qsort()` function. The result is written to a file. The name of the file is being read from the keyboard.
You will also need to implement a comparison function to make `qsort()` work correctly. If you do not know how to do this, just skip it, but still call the `qsort()` function.
You can assume that the `struct` above are part of your program, and that `data.txt` will never contain more than 30 entries.
*You may **not** use global variables.*
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.6 *A coffee machine* (6 points)

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`

To test the coffee machine, please write a program that just creates a **regular coffee with milk and sugar**. Then the order is replaced to have **cream** instead of **milk**.
Since all properties are just stored in the `char`, you need to come up with a way to efficiently store all properties.

Also implement (and use) the functions

```
set_coffee_poperty(struct coffee* c, int property)
unset_coffee_poperty(struct coffee* c, int 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.

Totalpoints: 33