TIC1001—Introduction to Coputing and Programming I
National University of Singapore
School of Continuing and Lifelong Education

Practical Examination 2

18 November 2017

Time allowed: 2 hours

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **two** questions. The time allowed for solving this test is **2 hours**.
3. The maximum score of this test is **20 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
5. While you are also provided with the templates `q1-template.c` and `q2-template.c` to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness locally on VS Code and not depend only on the provided test cases. Do ensure that you pasted your answers correctly by running the test cases at least once.
6. In case there are problems with Coursemology.org and we are not able to upload the answers to Coursemology.org, you will be required to name your file `q1-<student no>.c` and `q2-<student no>.c` where `<student no>` is your student number and leave the file in the `tic1001-pe1` folder on the Desktop. If your file is not named correctly, we will choose any .c file found at random.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology and correctly left in the desktop folder at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Practical Exam, or the parts that are not uploaded correctly.
8. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.

ALL THE BEST!

Question 1: [15 marks]

Temasek Railway Metro System (TRMS) has recently upgraded the signalling system on their subway lines.

In the new system, each rail line is divided into blocks of a unit length. Each station on the line will occupy a particular block. For example, in the figure below, Station A is on block 44 and Station B is on block 89.

Âă

Fixed speed trains. A train travelling at speed s covers exactly s blocks in each time unit. Suppose a time unit is the smallest divisible unit, that means the train cannot exactly cover a distance smaller than s.

For example, if the train in the figure above is travelling at speed $s = 2$, it will never be able to stop exactly at Station B if it leaves from Station A. Because from block 44, it will go to block 46 in the next time unit, then block 48, and so on, until block 88. But in the time unit it will overshoot to block 90.

However, at speed $s = 3$, the train will move from block 44 to 47, then 47 to 50, and eventually reach block 86 and will be able to stop exactly at block 89.

A. Write a function `int best_speed(int start, int end, int max_speed)` which takes in the block number of the starting and ending station, and the maximum speed which the train can travel. The function returns the maximum integer speed that the train can travel at and yet be able to exactly reach the destination station. [5 marks]

Sample Execution:

Function call	Return value
`best_speed(44, 89, 4)`	3
`best_speed(44, 89, 5)`	5
`best_speed(44, 89, 10)`	9

Variable Speed trains Suppose now the trains have to accelerate up to their max speed from 0 and decelerate back to 0 to stop. The acceleration and deceleration rate for a train is 1 unit speed per unit time, i.e., in one time unit, the speed and increase or decrease by 1.

Using the same example in the figure, if the maximum speed of the train is $s = 5$, then the following table shows the location and speed of the train over time:

Time	Block	Speed
0	44	0
1	44	1
2	45	2
3	47	3
4	50	4
5	54	5
6	59	5
7	64	5
8	69	5
9	74	5
10	79	4
11	83	3
12	86	2
13	88	1
14	89	0

At $t = 0$ the train's speed is $s = 0$. Thus, at $t = 1$ it is still at the same place, but its speed has increased to $s = 1$. So in $t = 2$ it has moved to the next block. Its speed then keep increasing until it accelerates up to its maximum speed.

Notice then that it has to start decelerating at $t = 10$ in order to be able to stop at the ending block.

B. Write a function `int stopping_distance(int speed)` that takes as input the current speed of the train, and returns the minimum number of blocks required for the train to come to a complete stop at the given speed. [5 marks]

Sample Executions:

Function call	Return value
`stopping_distance(4)`	10
`stopping_distance(5)`	15

C. Write a function `travel(int start, int end, int max_speed)` which takes in the block number of the starting and ending station, and the maximum speed which the train can travel. The function then prints the location and speed of the train for each time unit like in the table above. Each time unit is on a new line and the numbers are separated by a single space. [5 marks]

Sample Executions (on next page):

`travel(44, 89, 5);`

```
0 44 0
1 44 1
2 45 2
3 47 3
4 50 4
5 54 5
6 59 5
7 64 5
8 69 5
9 74 5
10 79 4
11 83 3
12 86 2
13 88 1
14 89 0
```

`travel(44, 89, 10);`

```
0 44 0
1 44 1
2 45 2
3 47 3
4 50 4
5 54 5
6 59 6
7 65 6
8 71 5
9 76 4
10 80 3
11 83 3
12 86 2
13 88 1
14 89 0
```

`travel(44, 89, 4);`

```
0 44 0
1 44 1
2 45 2
3 47 3
4 50 4
5 54 4
6 58 4
7 62 4
8 66 4
9 70 4
10 74 4
11 78 4
12 82 3
13 85 2
14 87 1
15 88 1
16 89 0
```

Question 2: Hello World! from space! [5 marks]

Physicist Ms. Nostradamusy feels that her long years of waiting is finally paying off! The large radar arrays she helped setup received a few mysterious messages from deep space. Unfortunately, the massages were mixed with some unwanted "background noise" probably due to atmosphere distortion. Help her to recover the messages!

Each deep space message is represented as a **character string**. Write a function `void filter(const char *original, const char threshold, char *passed, char *failed)` which separates (filters) the `original` string into two output strings:

1. `passed` string: Contains all characters in the original that is **larger or equal** to the `threshold` character in terms of ASCII value.
2. `failed` string: Contains all characters in the original that is **lesser** than the `threshold` character.

Sample runs

Original	Threshold	Passed	Failed
`"ABCDEF abcdef 1234!@#"`	`'D'`	`"DEFabcdef"`	`"ABC␣␣1234!@#"`
`"PE is FUN!"`	`' '`	`"PE is FUN!"`	`""`

Note that:

- The order of the characters in `passed` and `failed` strings should be the same as in the original.
- All characters of the original string should be in either `passed` or `failed`.
- It is possible to have **empty** string as an answer.

[5 marks]

Appendix: ASCII Table

ASCII Control Characters

Num	Sym	Description
0	NUL	Null char
1	SOH	Start of Heading
2	STX	Start of Text
3	ETX	End of Text
4	EOT	End of Transmission
5	ENQ	Enquiry
6	ACK	Acknowledgment
7	BEL	Bell
8	BS	Backspace
9	HT	Horizontal Tab
10	LF	Line Feed
11	VT	Vertical Tab
12	FF	Form Feed
13	CR	Carriage Return
14	SO	Shift Out / X-On
15	SI	Shift In / X-Off
16	DLE	Data Line Escape
17	DC1	Device Control 1
18	DC2	Device Control 2
19	DC3	Device Control 3
20	DC4	Device Control 4
21	NAK	Negative Acknowl
22	SYN	Synchronous Idle
23	ETB	End of Trans Block
24	CAN	Cancel
25	EM	End of Medium
26	SUB	Substitute
27	ESC	Escape
28	FS	File Separator
29	GS	Group Separator
30	RS	Record Separator
31	US	Unit Separator

ASCII Printable Characters

Num	Sym	Num	Sym	Num	Sym
32		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>

— E N D O F P A P E R —