TIC1002—Introduction to Coputing and Programming II
National University of Singapore
School of Continuing and Lifelong Education

Practical Examination 2

25 April 2020

Time allowed: 1 hour

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 **one** questions. The time allowed for solving this test is **1 hour**.
3. The maximum score of this test is **10 marks** with 3 additional bonus 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 provided with the template `pe2-template.cpp` 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 `<mat no>.py` where `<mat no>.py` is your matriculation number and send file directly to your invigilator via Microsoft Teams.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology (or correctly submitted via Microsoft Teams to your invigilator) 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: Trace Together [10 marks]

TraceTogether, safer together. Join 1,100,000 users in stopping the spread of COVID-19 through community-driven contact tracing.

Specification

In this exam, we will crudely simulate the workings of the TraceTogether app by modelling persons and periodically recording their GPS location at the point in time.

A person is assumed to remain at the given GPS position from the time of the log, until and inclusive of the time of the next log. Thus, it might seem that the person is in two places at the same time, but this is a reasonable assumption as we want to be more relaxed in the constrains.

We can then query a person's record to determine the time he was at a GPS location, or whether two persons had been in close contact with each other.

Illustration

Suppose a GPS location is abstracted as single number, and two GPS positions are considered to be in close contact with each other when they differ by ± 10.

Below is a sample of the records belonging to four persons, P, Q, R and S:

Person P

Time	Position
100	100
200	200
300	400
500	200

Person Q

Time	Position
80	300
150	210
240	500

Person R

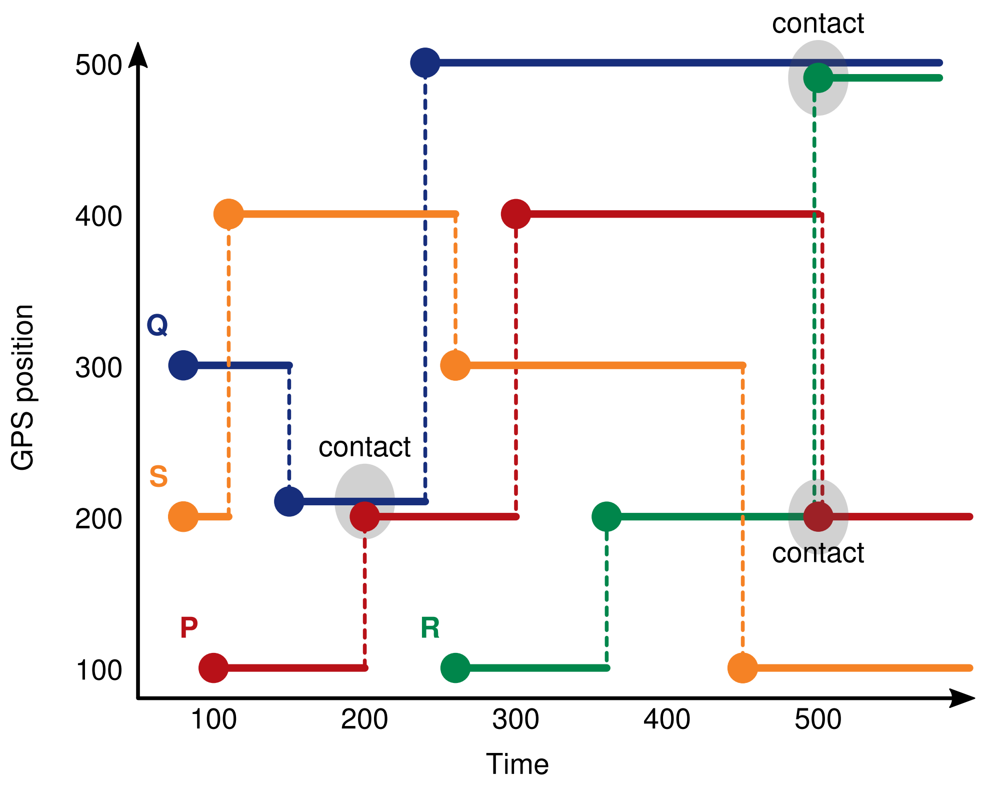
Time	Position
260	100
360	200
500	490

Person S

Time	Position
80	200
110	400
260	300
450	100

Close contact at a position Consider the record of Person P. Suppose we wish to query a time when P was near GPS position 190. From the record, we see that P has been recorded to be at position 200 during time 200–300 inclusive, and 500 and beyond. Since position 200 is within ± 10 of position 190, it is considered close contact during the aforementioned times.

Close contact with another person To better visualize the movement, we can plot the records of all four persons, P, Q, R and S in a graph as shown:



The dots on the graph represents each entry in each person's record.

- We can now easily see that person P had close contact with Q at time 200.
- At time 500, P just moved to position 200 while R moved from position 200. Due to the given assumption, we consider R to be at position 200 simultaneously as position 490. Thus, this to be a close contact between P and R.
- At the same time 500, R is also in close contact with Q.
- Although S has visited positions that are close with every one, they were all during a different time period. Thus, S had no close contact with any other persons.

Implementation

Advice: You should read through all parts and sample execution to get a better understanding of the overall expectations before attempting.

Provided ADTs

Assume that a GPS location is given as a `GPS` type. The function `bool close_contact` takes as inputs two `GPS` and returns `true` if the two locations are close enough to make contact, and `false` otherwise.

Traditionally, a GPS location stores the latitude, longitude and altitude. A rudimentary representation is provided in the template but you should not make any assumptions about its implementation. It will certainly not be defined the same way in Coursemology.

Your Tasks

Provide an implementation for the following struct and functions. Note that if you reuse any function that is to be implemented in another function, we will assume by function abstraction that the reused function works as specified, i.e. implementation error in the reused function will not carry forward.

A. `struct Person`

Design and implement `Person` as a `struct` to represent the state of a Person. You are free to incorporate any C++ STL type or implement your own ADTs.

Note that your choice of implementation might affect the efficiency of the latter functions.
[2 marks]

B. `void log(Person &p, long time, GPS position)`

The function takes as inputs a person ADT, a given time and a GPS position. It records in the ADT that the person is at the given position from the given time. Note that time can be thought of as seconds elapsed from some arbitrary time, and the function might not be called with increasing time. [2 marks]

C. `bool has_contact(Person &p, long time, GPS position)`

The function takes as inputs a person ADT, a given time and a GPS position. It returns `true` if that the person was ever in close contact with the given GPS position at the given time, otherwise `false` is returned. [3 marks]

D. `bool has_contact(Person &p, Person &q)`

The function takes two persons ADT as input and returns `true` if the two persons have been in close contact with each other according to their records, and `false` otherwise. [3 marks]

Bonus

The naive algorithm for part D `has_contact` runs in quadratic time. You can an additional 3 marks bonus if your implementation runs in linear time with respect to the inputs. That is you can potentially score 13 out of 10 marks for the PE2.

Sample Execution

The following code constructs the scenario illustrated above:

```
Person p;
log(p, 100, {100});
log(p, 500, {200});  // Note the order of logs need not be
log(p, 200, {200});  // in increasing time
log(p, 300, {400});
```

```
Person q;
log(q, 80,  {300});
log(q, 150, {210});
log(q, 240, {500});

Person r;
log(r, 260, {100});
log(r, 360, {200});
log(r, 500, {490});

Person s:
log(s, 80, {200});
log(s, 110, {400});
log(s, 260, {300});
log(s, 450, {100});
```

Function call	Return value
`has_contact(p, 250, {190})`	`true`
`has_contact(p, 400, {190})`	`false`
`has_contact(p, 700, {190})`	`true`
`has_contact(p, q)`	`true`
`has_contact(q, p)`	`true`
`has_contact(p, r)`	`true`
`has_contact(q, r)`	`true`
`has_contact(s, p)`	`false`
`has_contact(s, q)`	`false`
`has_contact(s, r)`	`false`

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 —