Computation I 5EIA1
C Exam Part 1: COVID Tracing (v1.2, October 23, 2020)
27 October 13:30-15:10

In this exam you will develop a COVID contact-tracing program. You can add and remove persons and contacts.

Important

- In this exam a predefined function is available for each task, e.g. `predefined_add_person`. Therefore, if you get stuck on a task or want to skip it then you can use the predefined function instead of your own function.
- If you use the predefined function for a task anywhere in your code then you will not get points for all of the test cases of that task. For example, if instead of writing your own `add_person` in Task 1 you use `predefined_add_person` in later tasks then you will not get the points for test cases 2 and 3 that are unique for Task 1. You will get points for the other test cases that use `predefined_add_person`.
- Oncourse will show all the test cases passed, including those using the predefined functions. Points for using predefined functions will be deducted after the exam.
- To use the predefined functions you need to include `#include "predefined.h"` in your program. This is done automatically when you create a new .c file.

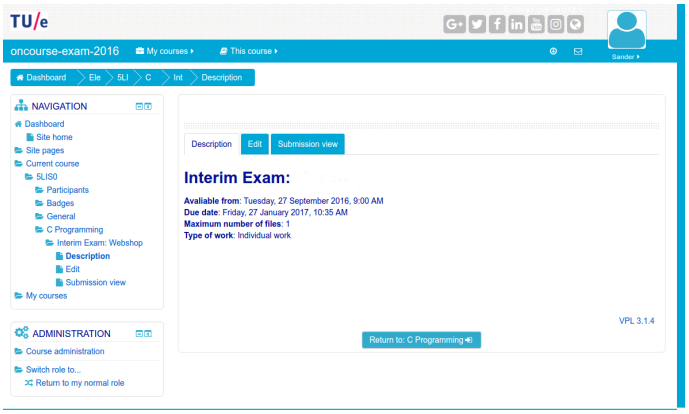
function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	% per fn	cumulative %
quit	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7%	7%
add person		1	1	1	1	1	1	1	1	1	1	1	1	1	21%	29%
print persons					1	1	1	1	1	1	1	1	1	1	14%	43%
add contact							1	1	1	1	1	1	1	1	14%	57%
remove contact									1	1	1		1	1	21%	79%
remove person												1	1	1	21%	100%
															100%	

Figure 1: Test cases and points per task. You can use predefined functions to not implement a function yourself but you will not get the points for that function. All test cases are listed at the end of this document.

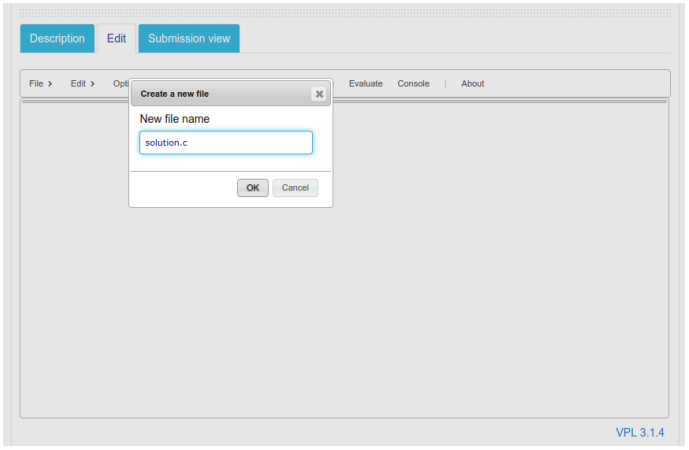
Important

- Your grade is based on the number of cases passed. Try to complete the cases one by one.
- You can press "evaluate" as often as you like during the exam to evaluate your solution. We advise you to do this regularly during the exam.
- You must follow the instructions of the exam. For example, you may not use an array if a linked-list implementation is asked for.
- You are only allowed to use materials offered electronically as part of the exam (cheat sheet and K&R book) during the exam. No other electronic or printed material are allowed.
- The grade is based on your **last submission**. Make sure you submit a working version that completes as many cases as possible. It's also wise not to make last-minute changes.
- Your program must work for all inputs, not just the test cases. For example, when a `#define MAX 10` must be defined as part of the exam, then your program should work for all values of `MAX`. We will change the test cases after the exam (but the number of test cases per task, i.e., the grade of correct programs will not change).

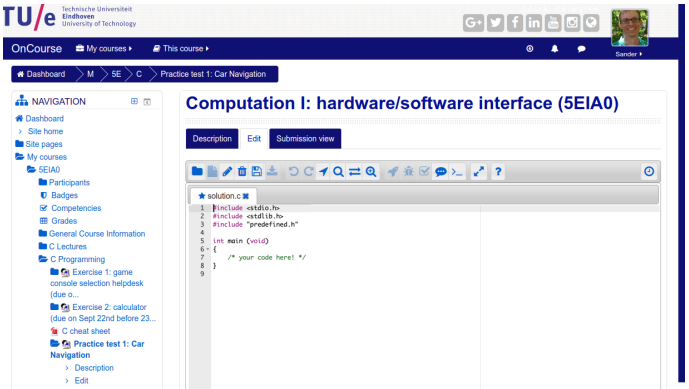
Open the correct C programming exam on exam.oncourse.tue.nl. You will see a screen similar to:



Select the "Edit" tab and the following screen will appear:



Provide a useful name for your C file (e.g., solution.c) and press "Ok". **The file name must end in .c and cannot contain any spaces.** You will then see the following screen:



You can write your own C program in the text editor that is now shown in your browser. Once you press "Save" you can "Run" and "Evaluate" your program. Using the "Run" command you will see a terminal where you can provide input to your program. You can use this to debug your program. Using the "Evaluate" command all test cases are evaluated and at the end the results are displayed (grade and errors if present).

Task 1. Write a C program that asks the user to select the command that needs to be performed. The commands that need to be supported are listed in the following table:

Character	Command
q	quit
a	add person
p	print the list of all persons
c	add a contact
r	remove a contact
R	remove a person

In this task you only need to implement the quit command. If an invalid character is given then print the error message `Unknown command 'x'` (with x replaced by the unknown command):

```
Command? z
Unknown command 'z'.
Command? q
Bye!
```

Your program must produce the *exact* output, including all spaces, capitalisation, quotes, etc.

Your program should now pass test case 1.

The `predefined.h` file that you should include in your program with `#include "predefined.h"` contains the following declarations:

```
struct person {
  char *name;
  struct person *contacts;
  struct person *next;
};

extern void predefined_add_person(struct person **persons, char *name);
extern void predefined_print_persons(struct person *persons);
extern struct person *predefined_find_person(struct person *persons, char *name);
extern void predefined_add_contact(struct person *person, char *name);
extern void predefined_remove_contact(struct person *person, char *name);
extern struct person *predefined_remove_person(struct person *persons, char *name);
```

Task 2.

In your main function define a variable `persons` of type pointer to person to keep track of all persons in a linked list. Add the 'a' command to add a new person. Write the function `add_person(struct person **persons, char *name)` that adds a new person with the given name at the *start* of the persons list. The `next` field in the struct is used for the list of persons. The `contacts` field will be used later; initialise it to `NULL`. Person names do not contain spaces; in other words, you can use the `"%s"` format string for `scanf`.

```
Command? a
Person? Hoest
Command? a
Person? Proest
Command? q
Bye!
```

Your program should now pass test cases 1 to 4. If you do not wish to implement the function `add_person` then you can use the `predefined_add_person` function, but you will not get the points for test cases 2-4. You can call the predefined function just like your own function:
`predefined_add_person(&persons, name);`

Hint: If when you run your program you receive an error message similar to:

```
/tmp/ccJmRCOb.o: In function ‘predefined_add_person’:
  exam.c:(.text+0x0): multiple definition of ‘predefined_add_person’
```

then you have named your own function `predefined_add_person` instead of `add_person`. When you write your own function then it should not contain `predefined_` in the name.

Task 3. Add the 'p' command to print all persons with the `void print_persons(struct person *persons)` function. The output below also shows how the contacts of each person should be printed. Contacts will be added in later tasks.

```
Command? p
Command? a
Person? Hoest
Command? p
Person: Hoest
Command? a
Person? Proest
Command? p
Person: Proest
Person: Hoest
Command? a
Person? Lach
Command? p
Person: Lach
Person: Proest
Person: Hoest
... insert some contacts - see later tasks ...
Command? p
Person: Lach
Person: Proest
- Hoest
- Lach
Person: Hoest
- Lach
Command? q
Bye!
```

Your program should now pass test cases 1 to 6. If you do not wish to implement the function `print_persons` then you can use the `predefined_print_persons` function, but you will not get the points for test cases 5-6.

Task 4. Add the 'c' command to add a contact with name to a person. Write two functions to do this:

```
struct person *find_person(struct person *persons, char *name)
void add_contact(struct person *person, char *name)
```

After asking for person A's name and person B's name, find person A in the persons list and then call `add_contact` with a pointer to person A and person B's name. Add person B to the *end* of person A's contact list.

The list of persons is a linked list made up of `struct persons` that are linked with the `next` field. Similarly, the list of contacts is a linked list made up of `struct persons` but they are linked with the `contacts` field (and the `next` field is `NULL`). The figure below illustrates this. The commands to generate this example are shown on the next page.

Note that a contact only works in one direction. So, when you add the contact Wheeze to the person Sneeze, then Wheeze is in Sneeze's contact list, but not the other way around. This is shown in the figure. Bidirectional contacts require two calls of the `add_contact` function.

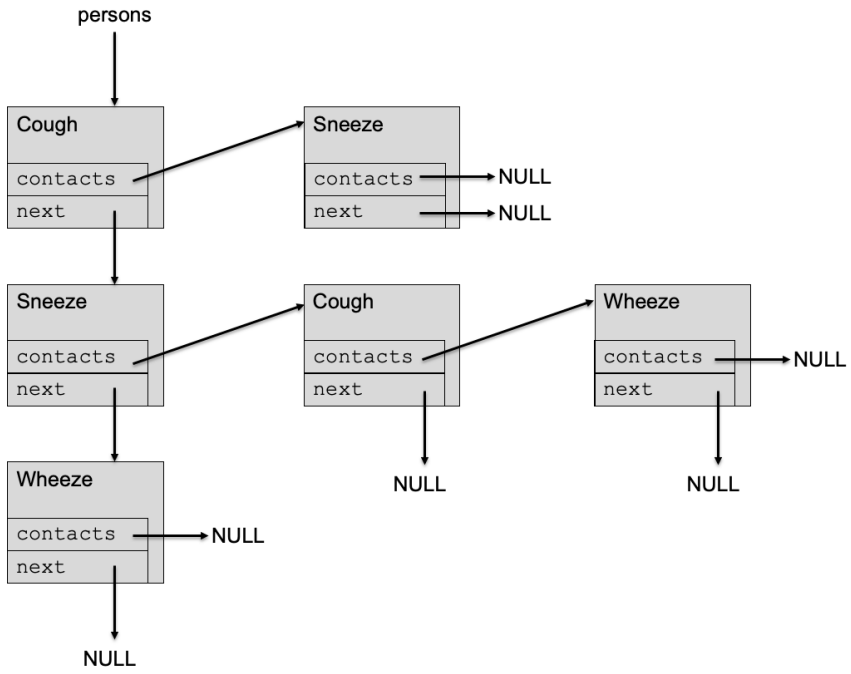


Figure 2: Persons and their contacts are both linked lists, using `next` and `contacts` fields respectively.

Your program should now pass test cases 1 to 8. If you do not wish to implement the function `add_contact` then you can use the `predefined_add_contact` function, but you will not get the points for test cases 7-8.

```
Command? a
Person? Wheeze
Command? a
Person? Sneeze
Command? a
Person? Cough
Command? p
Person: Cough
Person: Sneeze
Person: Wheeze
Command? c
Person A? Cough
Person B? Sneeze
Command? p
Person: Cough
- Sneeze
Person: Sneeze
Person: Wheeze
Command? c
Person A? Sneeze
Person B? Cough
Command? c
Person A? Sneeze
Person B? Wheeze
Command? p
Person: Cough
- Sneeze
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? q
Bye!
```

Task 5. Add the 'r' command to remove a contact from a person's contact list with the `void remove_contact(struct person *person, char *name)` function. You must also free any space that has been previously malloc'd.

Recall that contacts are one-way. Thus, after adding person Sneeze as a contact to person Cough, and after adding person Cough as contact to person Sneeze, then removing contact Sneeze from person Cough will leave person Cough as contact for person Sneeze. This is illustrated below.

```
... insert persons & contacts as on the previous page ...
Command? p
Person: Cough
- Sneeze
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? r
Person A? Cough
Person B? Sneeze
Command? p
Person: Cough
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? q
Bye!
```

Your program should now pass test cases 1 to 11. If you do not wish to implement the function `remove_contact` then you can use the `predefined_remove_contact` function, but you will not get the points for test cases 9-11.

Task 6. Add the 'R' command to remove a person from the list with the `struct person *remove_person(struct person *persons, char *name)` function. Remember to remove all the person's contacts first. Free all space that has been previously malloc'd.

In the example below note that when Plague is removed it must be removed from everyone else's contact list too.

```
Command? a
Person? Plague
Command? a
Person? COVID
Command? c
Person A? Plague
Person B? COVID
Command? c
Person A? COVID
Person B? Plague
Command? p
Person: COVID
- Plague
Person: Plague
- COVID
Command? R
Person? Plague
Command? p
Person: COVID
Command? q
Bye!
```

Your program should now pass test cases 1 to 14. If you do not wish to implement the function `remove_person` then you can use the `predefined_remove_person` function, but you will not get the points for test cases 12-14.

Submission: Your *last* submission will be graded. Note that points will be deducted after the exam for using predefined functions.

Input / output test cases

Long lines have been wrapped at 70 characters for legibility. When your program output is compared to the expected output lines will not be wrapped.

Case 01

Input:

```
q
```

Output:

```
Command? Bye!
```

Case 02

Input:

```
a
Hoest
a
Proest
q
```

Output:

```
Command? Person? Command? Person? Command? Bye!
```

Case 03

Input:

```
a
this-is-a-long-name
a
short
a
foo
a
bar
a
foobar
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Bye!
```

Case 04

Input:

```
a
Jip
a
Janneke
a
Kwik
a
Kwek
a
Kwak
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Bye!
```

Case 05

Input:

```
a
Jip
p
a
Janneke
p
a
Kwik
p
a
Kwek
p
a
Kwak
p
q
```

Output:

```
Command? Person? Command? Person: Jip
Command? Person? Command? Person: Janneke
Person: Jip
Command? Person? Command? Person: Kwik
Person: Janneke
Person: Jip
Command? Person? Command? Person: Kwek
Person: Kwik
Person: Janneke
Person: Jip
Command? Person? Command? Person: Kwak
Person: Kwek
Person: Kwik
Person: Janneke
Person: Jip
Command? Bye!
```

Case 06

Input:

```
a
Person1
a
Person11
a
Person111
a
Person1111
a
Person11111
a
Person111111
a
Person1111111
a
Person11111111
a
Person2
a
Person22
a
Person222
a
Person2222
a
Person22222
a
Person222222
a
Person2222222
a
Person22222222
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person: Person22222222
Person: Person2222222
Person: Person222222
Person: Person22222
Person: Person2222
Person: Person222
Person: Person22
Person: Person2
Person: Person11111111
Person: Person1111111
Person: Person111111
Person: Person11111
Person: Person1111
Person: Person111
Person: Person11
Person: Person1
Command? Bye!
```

Case 07

Input:

```
a
Wheeze
a
Sneeze
a
Cough
p
c
Cough
Sneeze
p
c
Sneeze
Cough
c
Sneeze
Wheeze
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person:
Cough
Person: Sneeze
Person: Wheeze
Command? Person A? Person B? Command? Person: Cough
- Sneeze
Person: Sneeze
Person: Wheeze
Command? Person A? Person B? Command? Person A? Person B? Command?
Person: Cough
- Sneeze
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? Bye!
```

Case 08

Input:

```
a
one
a
two
a
three
a
four
a
five
a
six
p
c
one
two
c
one
three
c
one
four
c
one
five
c
one
six
c
two
three
c
three
four
c
four
five
c
five
six
c
six
six
c
six
five
c
six
four
c
six
three
c
six
two
c
six
one
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Person? Command? Person: six
Person: five
Person: four
Person: three
Person: two
Person: one
Command? Person A? Person B? Command? Person A? Person B? Command?
Person A? Person B? Command? Person A? Person B? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person A? Person B? Command? Person A? Person B? Command?
Person A? Person B? Command? Person A? Person B? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person A? Person B? Command? Person: six
- six
- five
- four
- three
- two
- one
Person: five
- six
Person: four
- five
Person: three
- four
Person: two
- three
Person: one
- two
- three
- four
- five
- six
Command? Bye!
```

Case 09

Input:

```
a
Wheeze
a
Sneeze
a
Cough
c
Cough
Sneeze
c
Sneeze
Cough
c
Sneeze
Wheeze
p
r
Cough
Sneeze
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person: Cough
- Sneeze
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? Person A? Person B? Command? Person: Cough
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? Bye!
```

Case 10

Input:

```
a
Wheeze
a
Sneeze
a
Cough
c
Cough
Sneeze
c
Sneeze
Cough
c
Sneeze
Wheeze
p
r
Cough
Sneeze
p
r
Sneeze
Wheeze
r
Sneeze
Cough
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person: Cough
- Sneeze
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? Person A? Person B? Command? Person: Cough
Person: Sneeze
- Cough
- Wheeze
Person: Wheeze
Command? Person A? Person B? Command? Person A? Person B? Command?
Person: Cough
Person: Sneeze
Person: Wheeze
Command? Bye!
```

Case 11

Input:

```
a
Wheeze
a
Sneeze
a
Cough
c
Wheeze
Wheeze
c
Cough
Sneeze
c
Cough
Cough
c
Sneeze
Cough
c
Sneeze
Wheeze
c
Sneeze
Sneeze
p
r
Sneeze
Wheeze
p
r
Sneeze
Cough
p
r
Sneeze
Sneeze
p
r
Cough
Cough
r
Cough
Sneeze
p
r
Wheeze
Wheeze
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person A? Person B? Command? Person A? Person B? Command?
Person A? Person B? Command? Person: Cough
- Sneeze
- Cough
Person: Sneeze
- Cough
- Wheeze
- Sneeze
Person: Wheeze
- Wheeze
Command? Person A? Person B? Command? Person: Cough
- Sneeze
- Cough
Person: Sneeze
- Cough
- Sneeze
Person: Wheeze
- Wheeze
Command? Person A? Person B? Command? Person: Cough
- Sneeze
- Cough
Person: Sneeze
- Sneeze
Person: Wheeze
- Wheeze
Command? Person A? Person B? Command? Person: Cough
- Sneeze
- Cough
Person: Sneeze
Person: Wheeze
- Wheeze
Command? Person A? Person B? Command? Person A? Person B? Command?
Person: Cough
Person: Sneeze
Person: Wheeze
- Wheeze
Command? Person A? Person B? Command? Person: Cough
Person: Sneeze
Person: Wheeze
Command? Bye!
```

Case 12

Input:

```
a
Plague
a
COVID
a
SARS
a
Flu
p
R
Plague
p
R
SARS
p
R
Flu
p
R
COVID
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person: Flu
Person: SARS
Person: COVID
Person: Plague
Command? Person? Command? Person: Flu
Person: SARS
Person: COVID
Command? Person? Command? Person: Flu
Person: COVID
Command? Person? Command? Person: COVID
Command? Person? Command? Command? Bye!
```

Case 13

Input:

```
a
Plague
a
COVID
c
Plague
COVID
c
COVID
Plague
p
R
Plague
p
R
COVID
p
q
```

Output:

```
Command? Person? Command? Person? Command? Person A? Person B?
Command? Person A? Person B? Command? Person: COVID
- Plague
Person: Plague
- COVID
Command? Person? Command? Person: COVID
Command? Person? Command? Command? Bye!
```

Case 14

Input:

```
a
one
a
two
a
three
a
four
a
five
a
six
p
c
one
two
c
one
three
c
one
four
c
one
five
c
one
six
c
two
three
c
three
four
c
four
five
c
five
six
c
six
six
c
six
five
c
six
four
c
six
three
c
six
two
c
six
one
p
R
four
```

Output:

```
Command? Person? Command? Person? Command? Person? Command? Person?
Command? Person? Command? Person? Command? Person: six
Person: five
Person: four
Person: three
Person: two
Person: one
Command? Person A? Person B? Command? Person A? Person B? Command?
Person A? Person B? Command? Person A? Person B? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person A? Person B? Command? Person A? Person B? Command?
Person A? Person B? Command? Person A? Person B? Command? Person A?
Person B? Command? Person A? Person B? Command? Person A? Person B?
Command? Person A? Person B? Command? Person: six
- six
- five
- four
- three
- two
- one
Person: five
- six
Person: four
- five
Person: three
- four
Person: two
- three
Person: one
- two
- three
- four
- five
- six
Command? Person? Command? Person: six
- six
- five
- three
- two
- one
Person: five
- six
Person: three
Person: two
- three
Person: one
- two
- three
- five
- six
Command? Person A? Person B? Command? Person? Command? Person: six
- six
- five
- three
- two
Person: five
- six
Person: three
Person: two
- three
Command? Person? Command? Person: six
- six
- five
```