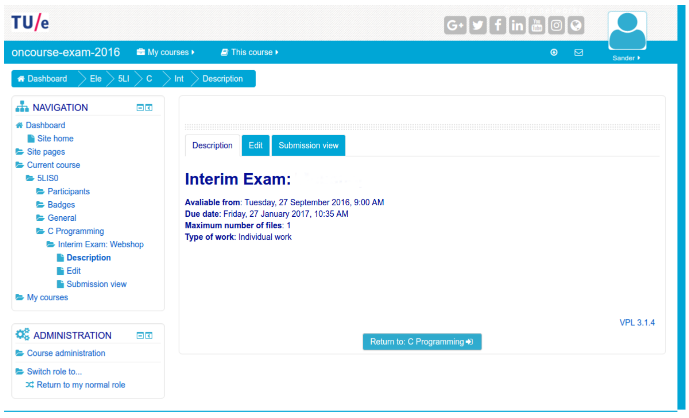
Interim Exam: Package Delivery
15-10-2018, 18.00-21.00

In this interim exam you will develop a program that can be used to compute the optimal (shortest) route between a series of houses where the delivery truck needs to drop-off parcels.

Important

- Your grade is based on the number of cases passed. So try to complete the cases one by one. Note that there may be additional test cases that are only run after the exam.
- 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 use 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 the Kernighan & Ritchie book as a reference during the exam. No other books or printed material are allowed.
- The grade is based on your **last submission**. So make sure you submit a working version that completes as many cases as possible.
- Your program must work for all inputs, not just the test cases. We will change the test cases after the exam (but the number of test cases per task, i.e., the grade will not change).

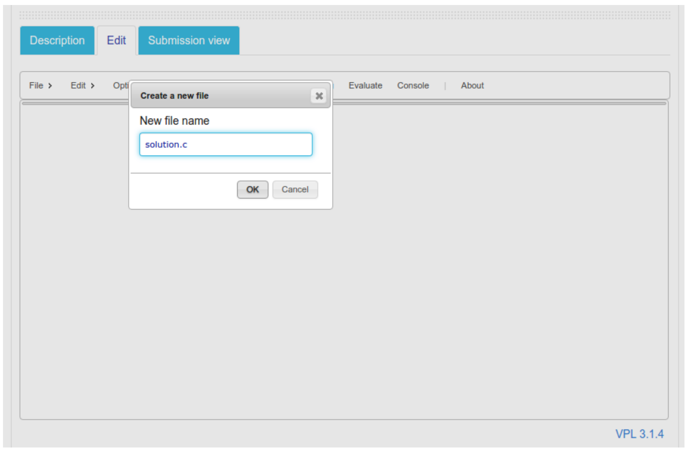
Task 1. Open the correct C programming exam on exam.oncourse.tue.nl. You will see a screen similar to:



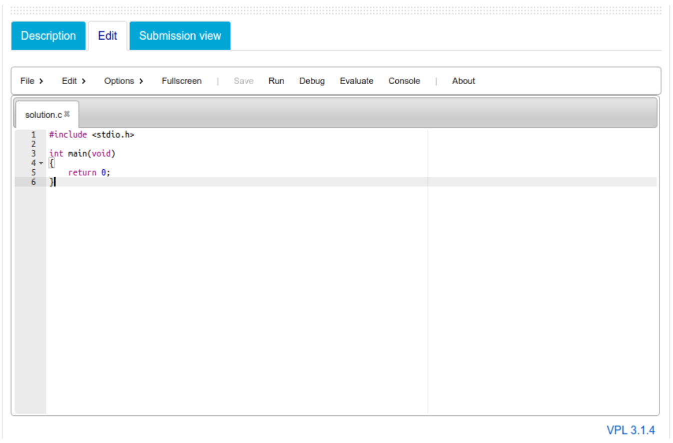
Interim Exam:

Avaliable from: Tuesday, 27 September 2016, 9:00 AM
Due date: Friday, 27 January 2017, 10:35 AM
Maximum number of files: 1
Type of work: Individual work

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". File names cannot contain spaces and must end in .c
You will then see the following screen:



```c
#include <stdio.h>

int main(void)
{
    return 0;
}
```

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 2. 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
a	add drop-off point
p	print list of all drop-off points
r	remove drop-off point
d	compute distance of entire route
w	swap two drop-off points on the route
s	sort drop-off points to minimize total route distance
q	quit

For the first step, implement the quit command. The output of your program should look as follows:

```
Command (a/p/r/d/w/s/q): q
```

Task 3. Create a struct of type "`struct location`" that can be used to store the x and y coordinate of a drop-off location. Both elements must be of type `float`. In addition, the struct should be able to store a label, i.e. name, with which the location can be identified), and a pointer to the next location in the list. Note that you should use a linked list to store all locations.

Task 4. Write a function "`struct location *addLocation(struct location *route, float x, float y, char *label)`" that adds a new location with the label `label` and supplied `x` and `y` coordinates at the end of the list `route`. The function should return a pointer to the start of the updated route.

If a location with the label `label` already exists in the list `route`, then the location should not be added again. Instead the following error message should be printed (replace `tjakka` with the actual label of the location):

```
Location with label tjakka already exists.
```

Extend the program such that when the user inputs the character 'a' as a command, the user is asked to input the label and coordinates of the location and subsequently this location is added to the route using the function `addLocation`. The output of your program should now look as follows:

```
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Location with label Den Dolech 2, Eindhoven already exists.
Command (a/p/r/d/w/s/q): a
Add Location
 x y label? 10.3 15.2 Groene loper 19, Eindhoven
Command (a/p/r/d/w/s/q): q
```

Task 5. Write a function "`void printRoute(struct location *route)`" that outputs all locations contained in the route. For each location, the label should be printed followed by the x and y coordinate (both with 1 digit precision).

Extend the program such that when the user inputs the character 'p' as a command, the function `printRoute` is executed. The output of your program should now look as follows:

```
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 13.35 15.02 Groene Loper 19, Eindhoven
Command (a/p/r/d/w/s/q): p
Den Dolech 2, Eindhoven: 0.0, 1.0
Groene Loper 19, Eindhoven: 13.4, 15.0
Command (a/p/r/d/w/s/q): q
```

Task 6. Write a function "`struct location *removeLocation(struct location *route, char *label)`" that removes the location with label `label` from the list `route`. The function should return a pointer to the updated list `route`.

Extend the program such that when the user inputs the character 'r' as a command, the user is asked to input the label of the location to be removed and subsequently this command is executed using the function `removeLocation`. The output of your program should now look as follows:

```
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 3 Groene Loper 19, Eindhoven
Command (a/p/r/d/w/s/q): r
Remove location
 label? Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): p
Groene Loper 19, Eindhoven: 0.0, 3.0
Command (a/p/r/d/w/s/q): q
```

If the supplied `label` does not exist on the route, then the following error message should be printed (replace `tjakka` with the actual label of the location):

```
Location with label tjakka does not exist on route.
```

The output of the program then looks as follows:

```
Command (a/p/r/d/w/s/q): r
Remove location
 label? Den Dolech 2, Eindhoven
Location with label Den Dolech 2, Eindhoven does not exist on route.
Command (a/p/r/d/w/s/q): q
```

Task 7. Write a function "`struct location *swapLocations(struct location *route, char *label1, char *label2)`" that swaps the locations with labels `label1` and `label2` in the list `route`. The function should return a pointer to the updated list `route`.

Extend the program such that when the user inputs the character 'w' as a command, the user is asked to input the labels of the locations to be swapped and subsequently this command is executed using the function `swapLocations`. The output of your program should now look as follows:

```
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 3 Groene Loper 19, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 5 Julianalaan 23, Tilburg
Command (a/p/r/d/w/s/q): w
Swap locations
 label 1? Den Dolech 2, Eindhoven
 label 2? Groene Loper 19, Eindhoven
Command (a/p/r/d/w/s/q): p
Groene Loper 19, Eindhoven: 0.0, 3.0
Den Dolech 2, Eindhoven: 0.0, 1.0
Julianalaan 23, Tilburg: 0.0, 5.0
Command (a/p/r/d/w/s/q): q
```

If the supplied `label` does not exist on the route, then the following error message should be printed (replace `tjakka` with the actual label of the location):

```
Location with label tjakka does not exist on route.
```

The output of your program should now look as follows:

```
Command (a/p/r/d/w/s/q): w
Swap location
 label 1? Den Dolech 2, Eindhoven
 label 2? Groene Loper 19, Eindhoven
Location with label Den Dolech 2, Eindhoven does not exist on route.
Command (a/p/r/d/w/s/q): q
```

Task 8. Write a function "`float computeDistanceBetweenLocations(struct location *a, struct location *b)`" that computes the distance between the locations `a` and `b`.

Consider the locations (x_1, y_1) and (x_2, y_2). Assuming that both locations are in the positive quadrant of the coordinate system (i.e., $x_1 \geq 0$, $x_2 \geq 0$, $y_1 \geq 0$, $y_2 \geq 0$), the distance between these locations is equal to:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Note however that (x_i, y_i) do not have to be positive. Hence you need to adapt the above formula in your implementation.
Hint: you can use the function "`float sqrt(float)`" to compute the square root. To use this function, you must include the file `math.h`.

Task 9. Write a function "`float computeDistanceOfRoute(struct location *route)`" that computes the sum of the distance between all subsequent locations on the route starting from the origin (i.e., coordinate $(0, 0)$) to the first location in the list and subsequently continuing till the last element in the list.

Extend the program such that when the user inputs the character 'd' as a command, the function `computeDistanceOfRoute` is used to compute the distance of the entire `route` assuming the distance from the origin $(0, 0)$ to the first location in the list to also be part of the route. The output of your program should now look as follows:

```
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 3 Groene Loper 19, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 3 0 De Zaale 34, Eindhoven
Command (a/p/r/d/w/s/q): d
Distance: 7.2
Command (a/p/r/d/w/s/q): q
```

Task 10. Write a function "`struct location *sortLocationsOnDistanceOfRoute(struct location *route)`" that sorts all locations in the list `route` on the distance to the last location in the sorted list. The very first location added in the sorted route should be the point closest to the origin $(0, 0)$. The function should return a pointer to the updated `route`.

Of course you will have to extend the set of operations supported by your program. You can use the command 's' for the operation 'sort'. The output of your program should then look as follows:

```
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 3 Julianalaan 23, Tilburg
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 0 1 Den Dolech 2, Eindhoven
Command (a/p/r/d/w/s/q): a
Add location
 x y label? 3 3 De Zaale 34, Eindhoven
Command (a/p/r/d/w/s/q): s
Sort route
Command (a/p/r/d/w/s/q): p
Den Dolech 2, Eindhoven: 0.0, 1.0
Julianalaan 23, Tilburg: 0.0, 3.0
De Zaale 34, Eindhoven: 3.0, 3.0
Command (a/p/r/d/w/s/q): q
```

Submission: Your final solution must be submitted through OnCourse which will automatically grade this submission. The input and output of the test cases used by OnCourse is not provided on the next page (due to the length of some cases). It will however as always appear when a test case fails on OnCourse.