UNIVERSITÄT DUISBURG ESSEN

NES Networked Embedded Systems

Programming in C/C++

Exercise Sheet 4
!!! Deadline: Tue, Nov 26th, 23:55 !!!

Task 01:
Calculate the maximum of two numbers in C++. Download the provided template code file (ex04task01.cpp) and complete it by providing the following implementation:

- Write a preprocessor macro `MAXM()`. Ensure that your implementation is as robust as possible, however, it is expected that it behaves strangely in some situations. Note that Doxygen does not provide a simple way to document macro functions. Simply treat the macro function as a regular function when documenting it.
- Write two overloaded functions `maxf()`, one accepting and returning int, the second accepting and returning double. When either function is called, first print out: "In `maxf(int)`" or "In `maxf(double)`" for the int and double versions respectively.
- Do **not** modify the `main()` function in any way!

Also write a report which answers the following questions:

- What are the advantages and disadvantages of the macro and function solutions?
- Why does the commented line in the code not compile?
- Which `maxf()` functions are called for the `maxf1-maxf3` outputs and why?

Submission: The executable generated by your Makefile should be called: **`macromax`**
Points: (code 10pts, report 10pts, comment 5pts)

Task 02:
Since the preprocessor basically replaces text, macros can also be used to generate code. Download the code template ex04task02.cpp and complete it by implementing a macro `MAKEVAR` that can be used like this

```
MAKEVAR(weight, int)
```

which is replaced by the preprocessor by

```
int weight;
void set_weight(int val) { weight = val; }
int get_weight(void) { return weight; }
```

Add a comment to your macro to describe how it works.
Do *not* modify the main() function in any way!!

Note: To make long preprocessor statements more readable you can end a line with a backslash and continue it on the next line. For example

```
#define LONG_TEXT "This is a " \
"long text to be replaced"
```

Submission: Note that the code is a cpp file when writing your Makefile. Your executable for this task should be called **`makevar`**
Points: (code 10pts, comment 5pts)

<u>Task 03</u>:
Given is the sample program listed below (can be downloaded as ex04task03.c).

- What is the output of the following program? What is the error in the program?
- Describe how you use gdb to find the error but without single steps (commands “n(next)” or “s(tep)”)! Include all gdb commands in your report.
- Assume that you have a bigger program which cannot just be inspected by manually checking the source code.

```
int main(void) {
  struct stest {
    int top;
    int arr[5];
    int bottom;
  } s;
  int i;

  s.top = 1;
  s.bottom = 2;

  for (i = 0; i <= 5; i++) {
    s.arr[i] = 42;
  }
  printf("top = %d, bottom = %d\n", s.top, s.bottom);
}
```

<u>Submission</u>: Submit the code provided and call your executable **`overflow`**. Also submit your report as a single answer.txt file
<u>Points</u>: (report 20pts)

<u>Task 04</u>:
Implement a class “`StopWatch`” that allows for measuring time (in seconds), which has the following methods:

- Method `void start(void)` starts the time measurement.
- Method `double stop(void)` stops the measurement returns the elapsed time since start.
- Put the class header and implementation in the same file, called `stopwatch.cpp`. This class will be used in future exercises

Both the `start()` and `stop()` methods shall print errors if they are not called in the correct order (which is start before stop). Use the following text: “StopWatch already running!” and “StopWatch not started yet!” respectively for the error messages.

Test your class with the sample code provided as template ex04task04.cpp. Do not modify the main() function in any way!

<u>Hint</u>: Look at the clock() function of header file “ctime”.
<u>Submission</u>: Include the test program ex04task04.cpp in your submission zip and use it in your Makefile to generate the executable, which should be called **`stopwatch`**

<u>Points</u>: (code 35 pts, comments 5 pts)