RADBOUD UNIVERSITEIT NIJMEGEN
AFDELING STERRENKUNDE
JÖRG R. HÖRANDEL
JAN VAN ROESTEL

Programming 1 – 2015/16
Computer Practicum 7 – 04.01.2016

The answers should be handed in before 11.01.2016 on blackboard!

Write as a comment on the first line your name and student number:
/* <name> <student number> */
#include <stdio.h>

Task 20 (4 points)
Write a program for a simple calculator. It should be able to execute simple operations: addition "+", subtraction "-", multiplication "*", and division "/" of two numbers. For example when you enter:
12/4
the program should return 3.

The program should read its input from the command line. Use *int main(int argc, char *argv[])* to access the command line parameters. Note: command line parameters are separated by space (" "). This means, depending on how the input is written it is stored in different ways in *argv[]*. E.g. "12/4" is ONE argument, i.e. one element in *argv*. "12 / 4" will be stored in three elements of the vector *argv[]*, since there are spaces (" ") in between the characters. If needed you need to combine the strings into one string in order to pass the arguments to the function *evaluate* described below.

Write a function *evaluate(char *text)*. This function uses a string <text> as input and returns the result of the operation. The function *evaluate* searches for the characters "+", "-", "*" and "/" in <text>. The characters before the operator are then assigned to a float variable and the characters after the operator to a second float variable. Use the function *sscanf*. After this is done, execute the corresponding calculation.

Print out the result of the operation in the main program.

Task 21 (5 points)
Write a program to fit a straight line using functions from "Numerical Recipes".

Write a text file "*data.dat*" (with an editor of your choice), which contains the following data (x values, y values, and uncertainties on y):

1 11.0 0.5
2 13.0 0.8
3 12.5 0.7
4 14.3 0.6
5 14.8 0.9
6 16.9 0.9
7 18.1 1.1
8 17.9 1.5
9 19.0 0.2

In your program, open the file (*data.dat*) and read the values. Call the function *void fit(float x[], float y[], int ndata, float sig[], int mwt, float *a, float *b, float *siga, float *sigb, float *chi2, float *q)* from the "Numerical Recipes" library. Note: we have y values with uncertainties. The function *fit* and a description of the parameters have been given in the lecture.

Write the result of the fit: parameters a and b as well as their uncertainties $siga$ and $sigb$ as floating point values into a file "*results.dat*".

Attach the contents of both files "*data.dat*" and "*results.dat*" as a comment at the end of your program and upload this to blackboard.

To use the functions from "Numerical Recipes" download the *tar file* from the course web site. Write the following lines at the begin of your code:

```
#include <stdio.h>
#include "gammln.c"
#include "gser.c"
#include "nrerror.c"
#include "gcf.c"
#include "gammq.c"
#include "fit.c"
```

#include "" includes a local file into your program.

Task 22 (3 points)
A leap day occurs every four years, with the exception of century years. These have only a leap day if they are dividable by 400 (so 1900 does not have a leap day, but 2000 does). Write a function *char *check_year(int year).* The function returns one of the two strings: "The year %i is a leap year" or "The year %i is not a leap year".

In the main program call the function *check_year* in a *for* loop and print out the result. The loop should print out the result for 20 subsequent years. The starting year is defined as follows: Take the last 4 digits of your student number as a year, i.e. s1234567 would correspond to the year 4567. For this example the *for* loop runs from 4567 to 4587.

Write the results for your student number as a comment at the end of the program and upload it to blackboard.

Task 23 (3 points)
Check the random number generator discussed during the lecture.

```
unsigned long next=1;

/* random number generator */
int rand(void)
 {
  next = next * 1103515245 + 12345;
  return(unsigned int) (next/65536) % 32768;
 }

/* set seed value */
void srand(unsigned int seed)
 {
  next = seed;
 }
```

Write a program that generates 300 random numbers between 0 and 1. Ideally they are distributed uniformly. Check this by making a simple histogram, i.e. write a program that generates output similar to the following one:

```
0.0 .. 0.1  ###################
0.1 .. 0.2  ######################
0.2 .. 0.3  ################
0.3 .. 0.4  ######################
0.4 .. 0.5  #############
0.5 .. 0.6  #################
0.6 .. 0.7  ##################
0.7 .. 0.8  ##############
0.8 .. 0.9  #################
0.9 .. 1.0  ######################
```

Each "#" represents a random number in the corresponding range.

Write the output as a comment at the end of your C code and upload it to blackboard.

Happy New Year!