Contents

1 README

```
guy.pelc
Guy Pelc (203552823)
EX: 1

FILES:
osm.cpp -- implementation of the osm library.
results.png -- graph of the results.
Makefile
README

REMARKS:
1) in the osm implementation, the number of iterations are rounded up by 50 due to the loop unrolling
2) the results in the graph were calculated by an average of 10^6 iterations.

ANSWERS:

Assignment 1:
WhatIDo:
-requires a single input, let us call it INPUT
-let us call my username USRNAME

1) creates a set of directories and a file:
/Welcome/To/OS2018

2) in the file OS2018 writes:
USRNAME\nIf you haven't read the course guidelines yet --- do it right now!\nINPUT

where in my case USRNAME=guy.pelc
and INPUT is the input i entered when running the function from the terminal.

3) closes the file OS2018, then removes the directories "To" and "Welcome"
```

2 Makefile

```
CC=g++
CXX=g++
RANLIB=ranlib

LIBSRC=osm.cpp
LIBOBJ=$(LIBSRC:.cpp=.o)

INCS=-I.
CFLAGS = -Wall -std=c++11 -g $(INCS)
CXXFLAGS = -Wall -std=c++11 -g $(INCS)

OSMLIB = libosm.a
TARGETS = $(OSMLIB)

TAR=tar
TARFLAGS=-cvf
TARNAME=ex1.tar
TARSRCS=$(LIBSRC) Makefile README

all: $(TARGETS)

$(TARGETS): $(LIBOBJ)
    $(AR) $(ARFLAGS) $@ $^
    $(RANLIB) $@

clean:
    $(RM) $(TARGETS) $(OSMLIB) $(OBJ) $(LIBOBJ) *~ *core

depend:
    makedepend -- $(CFLAGS) -- $(SRC) $(LIBSRC)

tar:
    $(TAR) $(TARFLAGS) $(TARNAME) $(TARSRCS)
```

3 osm.cpp

```cpp
#include "osm.h"
#include <iostream>
#include <sys/time.h>
using namespace std;

/* Initialization function that the user must call
 * before running any other library function.
 *
 * It is empty in this implementation, therefore returns 0 always.
 */
int osm_init(){
    return 0;
}

/* finalizer function that the user must call
 * after running any other library function.
 * The function may, for example, free memory or
 * close/delete files.
 * Returns 0 uppon success and -1 on failure
 *
 * It is empty in this implementation, therefore returns 0 always.
 */
int osm_finalizer(){
    return 0;
}

/* Time measurement function for a simple arithmetic operation, addition.
   returns time in nano-seconds upon success,
   and -1 upon failure.

   rounds up number of iterations by 50
   */
double osm_operation_time(unsigned int iterations){
    timeval t {},t2 {};

    if (iterations == 0) {iterations = 1000;}
    unsigned int j = 0;
    unsigned int a = 0;

        if (gettimeofday( &t, nullptr) == -1) {return -1;}
        while (j < iterations){


            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;

            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
            a = 1 + 1;
```

```
60              a = 1 + 1;
61              a = 1 + 1;
62              a = 1 + 1;
63              a = 1 + 1;
64              a = 1 + 1;
65
66              a = 1 + 1;
67              a = 1 + 1;
68              a = 1 + 1;
69              a = 1 + 1;
70              a = 1 + 1;
71              a = 1 + 1;
72              a = 1 + 1;
73              a = 1 + 1;
74              a = 1 + 1;
75              a = 1 + 1;
76
77              a = 1 + 1;
78              a = 1 + 1;
79              a = 1 + 1;
80              a = 1 + 1;
81              a = 1 + 1;
82              a = 1 + 1;
83              a = 1 + 1;
84              a = 1 + 1;
85              a = 1 + 1;
86              a = 1 + 1;
87
88              a = 1 + 1;
89              a = 1 + 1;
90              a = 1 + 1;
91              a = 1 + 1;
92              a = 1 + 1;
93              a = 1 + 1;
94              a = 1 + 1;
95              a = 1 + 1;
96              a = 1 + 1;
97              a = 1 + 1;
98
99              j += 50;
100         }
101         if (gettimeofday( &t2, nullptr) == -1) {return -1;}
102
103     if (a) {}
104
105
106     double diff = long(double(t2.tv_usec-t.tv_usec))*(1000./j);
107     return (diff >= 0 ? diff : -1);
108
109 }
110
111 /* a function that does nothing */
112 void empty_function(){}
113
114 /* Time measurement function for an empty function call.
115    returns time in nano-seconds upon success,
116    and -1 upon failure.
117
118    rounds up number of iterations by 50
119    */
120
121 double osm_function_time(unsigned int iterations){
122     timeval t {},t2 {};
123
124     if (iterations == 0) {iterations = 1000;}
125     unsigned int j = 0;
126
127     try {
```

```
128         if (gettimeofday( &t, nullptr) == -1) {return -1;} ;
129         while (j < iterations){
130 
131             empty_function();
132             empty_function();
133             empty_function();
134             empty_function();
135             empty_function();
136             empty_function();
137             empty_function();
138             empty_function();
139             empty_function();
140             empty_function();
141 
142             empty_function();
143             empty_function();
144             empty_function();
145             empty_function();
146             empty_function();
147             empty_function();
148             empty_function();
149             empty_function();
150             empty_function();
151             empty_function();
152 
153             empty_function();
154             empty_function();
155             empty_function();
156             empty_function();
157             empty_function();
158             empty_function();
159             empty_function();
160             empty_function();
161             empty_function();
162             empty_function();
163 
164             empty_function();
165             empty_function();
166             empty_function();
167             empty_function();
168             empty_function();
169             empty_function();
170             empty_function();
171             empty_function();
172             empty_function();
173             empty_function();
174 
175             empty_function();
176             empty_function();
177             empty_function();
178             empty_function();
179             empty_function();
180             empty_function();
181             empty_function();
182             empty_function();
183             empty_function();
184             empty_function();
185 
186             j += 50;
187         }
188         if (gettimeofday( &t2, nullptr)==-1){return -1;};
189     }
190     catch (exception &e){
191         return -1;
192     }
193 
194 
195     double diff = long(double(t2.tv_usec-t.tv_usec))*(1000./j);
```

```
196     return (diff >= 0 ? diff : -1);
197
198 }
199
200
201 /* Time measurement function for an empty trap into the operating system.
202    returns time in nano-seconds upon success,
203    and -1 upon failure.
204
205    rounds up number of iterations by 50
206    */
207 double osm_syscall_time(unsigned int iterations){
208     timeval t {},t2 {};
209
210     if (iterations == 0) {iterations = 1000;}
211     unsigned int j = 0;
212
213         if (gettimeofday( &t, nullptr) == -1) {return -1;};
214         while (j < iterations){
215
216             OSM_NULLSYSCALL;
217             OSM_NULLSYSCALL;
218             OSM_NULLSYSCALL;
219             OSM_NULLSYSCALL;
220             OSM_NULLSYSCALL;
221             OSM_NULLSYSCALL;
222             OSM_NULLSYSCALL;
223             OSM_NULLSYSCALL;
224             OSM_NULLSYSCALL;
225             OSM_NULLSYSCALL;
226
227             OSM_NULLSYSCALL;
228             OSM_NULLSYSCALL;
229             OSM_NULLSYSCALL;
230             OSM_NULLSYSCALL;
231             OSM_NULLSYSCALL;
232             OSM_NULLSYSCALL;
233             OSM_NULLSYSCALL;
234             OSM_NULLSYSCALL;
235             OSM_NULLSYSCALL;
236             OSM_NULLSYSCALL;
237
238             OSM_NULLSYSCALL;
239             OSM_NULLSYSCALL;
240             OSM_NULLSYSCALL;
241             OSM_NULLSYSCALL;
242             OSM_NULLSYSCALL;
243             OSM_NULLSYSCALL;
244             OSM_NULLSYSCALL;
245             OSM_NULLSYSCALL;
246             OSM_NULLSYSCALL;
247             OSM_NULLSYSCALL;
248
249             OSM_NULLSYSCALL;
250             OSM_NULLSYSCALL;
251             OSM_NULLSYSCALL;
252             OSM_NULLSYSCALL;
253             OSM_NULLSYSCALL;
254             OSM_NULLSYSCALL;
255             OSM_NULLSYSCALL;
256             OSM_NULLSYSCALL;
257             OSM_NULLSYSCALL;
258             OSM_NULLSYSCALL;
259
260             OSM_NULLSYSCALL;
261             OSM_NULLSYSCALL;
262             OSM_NULLSYSCALL;
263             OSM_NULLSYSCALL;
```

```
264             OSM_NULLSYSCALL;
265             OSM_NULLSYSCALL;
266             OSM_NULLSYSCALL;
267             OSM_NULLSYSCALL;
268             OSM_NULLSYSCALL;
269             OSM_NULLSYSCALL;
270 
271             j += 50;
272         }
273         if (gettimeofday( &t2, nullptr) == -1){return -1;};
274 
275 
276     double diff = long(double(t2.tv_usec-t.tv_usec))*(1000./j);
277     return (diff >= 0 ? diff : -1);
278 }
279 
280 
```

4 results.png