```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
  int freq[MAXPAROLA] ; /* vettore di contatori
  delle frequenze delle lunghezze delle parole */
  char riga[MAXRIGA] ;
  int i, inizio, lunghezza ;
  FILE * f ;

  for(i=0; i<MAXPAROLA; i++)
    freq[i]=0 ;

  if(argc != 2)
  {
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
  }
  f = fopen(argv[1], "rt") ;
  if(f==NULL)
  {
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
  }

  while( fgets( riga, MAXRIGA, f ) != NULL )
```

Algorithms and Programming

Introduction to Algorithms and Programming

Stefano Quer
Department of Control and Computer Engineering
Politecnico di Torino

Course Introduction

- Program syllabus
  - Algorithms and Programming
    - 02OGD$_{LM}$ & 01OGD$_{LP}$, ING-INF/05
  - Bachelor-level degree
    - Computer Engineering
      - 2° year, 12 credits, **120** hours
      - All students (from A to Z)
    - Electronic and Communication Engineering
      - 3° year, 10 credits, **100** hours
      - All students (from A to Z)

!!!

- Teacher
  - Quer Stefano

Course Introduction

- Laboratory assistants
  - Fabrizio Finocchiaro

  - Marco Palena

Subject Fundamentals

- Acquire **adequate** knowledge and skills in
  - Algorithms and data structures
  - Their implementation in C to solve complex problems
- Student should gradually evolve from more analytic to more design-oriented skills
- The course introduce
  - Theoretical foundations and solutions to "classical" problems
  - Advanced aspects in C language and problem-solving

Learning Outcomes

- Knowledge of
  - Advance C and modular programming
  - Dynamic memory allocation and use of pointers
  - Complexity analysis
  - Sorting algorithms
  - Recursion and recursive programming
  - Greedy problem-solving paradigms
  - Complex data structures and Abstract Data Types
    - Linked lists, queues, stacks, trees, binary search trees, hash tables, heaps, graphs

Prerequisites

- Incremental nature of the course with respect to the first year class "**Computer Science**"
- **Strict** prerequisites in terms of programming skills and programming language knowledge
  - Elementary computer systems architecture
  - Numeric systems, numbers and types
  - Syntax of C, basic data types and basic constructs
  - Basic programming skills in C for elementary problem solving

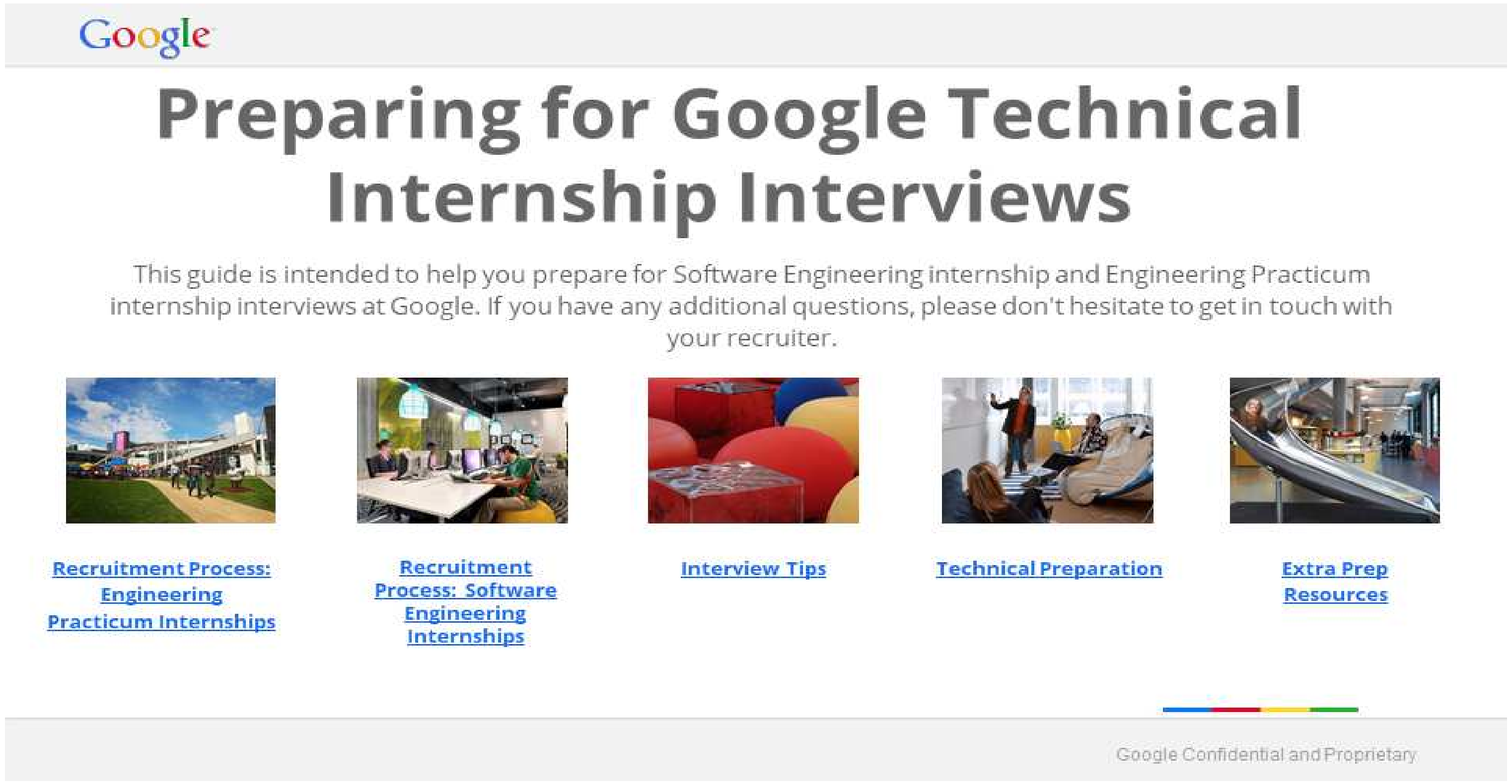
Contents

- Main course items
  - Review of basic language and problem solving
  - Algorithm analysis
  - Sorting algorithms
  - Static and dynamic data structures
  - Modularity and modular implementation
  - Recursion and recursive programs
  - Abstract objects, collections of objects and ADTs
  - Data structures for symbol tables
  - Graph theory
  - Problem solving

ECE students will be told **when** not to come to lessons (mainly in the second part)

Contents

- Doubts on the course impact within your curricula?



Google

Preparing for Google Technical Internship Interviews

This guide is intended to help you prepare for Software Engineering internship and Engineering Practicum internship interviews at Google. If you have any additional questions, please don't hesitate to get in touch with your recruiter.

Recruitment Process: Engineering Practicum Internships

Recruitment Process: Software Engineering Internships

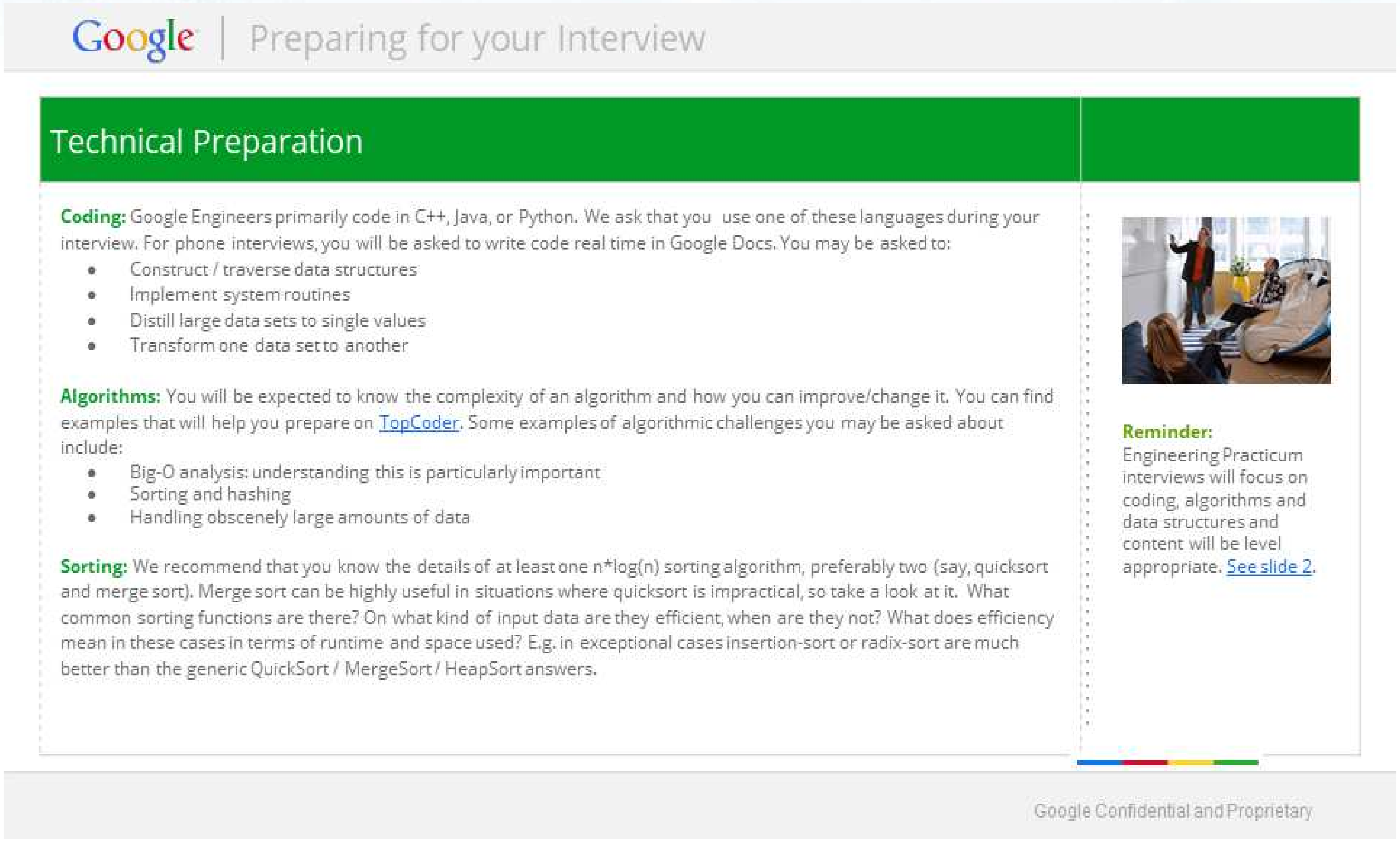
Interview Tips

Technical Preparation

Extra Prep Resources

Google Confidential and Proprietary

Contents



Google | Preparing for your Interview

Technical Preparation

Coding: Google Engineers primarily code in C++, Java, or Python. We ask that you use one of these languages during your interview. For phone interviews, you will be asked to write code real time in Google Docs. You may be asked to:

- Construct / traverse data structures
- Implement system routines
- Distill large data sets to single values
- Transform one data set to another

Algorithms: You will be expected to know the complexity of an algorithm and how you can improve/change it. You can find examples that will help you prepare on TopCoder. Some examples of algorithmic challenges you may be asked about include:

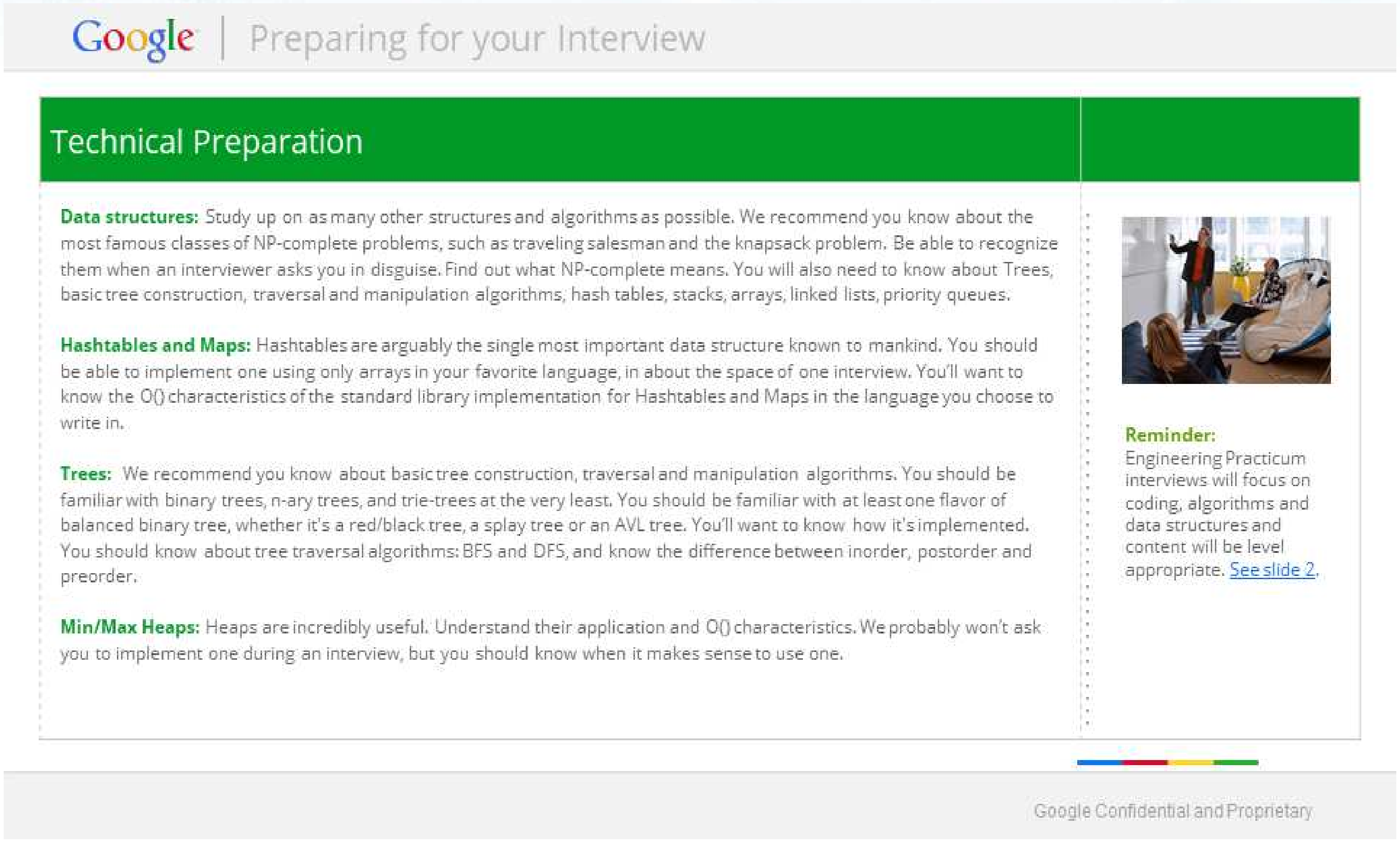
- Big-O analysis: understanding this is particularly important
- Sorting and hashing
- Handling obscenely large amounts of data

Sorting: We recommend that you know the details of at least one n*log(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it. What common sorting functions are there? On what kind of input data are they efficient, when are they not? What does efficiency mean in these cases in terms of runtime and space used? E.g. in exceptional cases insertion-sort or radix-sort are much better than the generic QuickSort / MergeSort / HeapSort answers.

Reminder:
Engineering Practicum interviews will focus on coding, algorithms and data structures and content will be level appropriate. See slide 2.

Google Confidential and Proprietary

Contents



Google | Preparing for your Interview

Technical Preparation

Data structures: Study up on as many other structures and algorithms as possible. We recommend you know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem. Be able to recognize them when an interviewer asks you in disguise. Find out what NP-complete means. You will also need to know about Trees, basic tree construction, traversal and manipulation algorithms, hash tables, stacks, arrays, linked lists, priority queues.

Hashtables and Maps: Hashtables are arguably the single most important data structure known to mankind. You should be able to implement one using only arrays in your favorite language, in about the space of one interview. You'll want to know the O() characteristics of the standard library implementation for Hashtables and Maps in the language you choose to write in.

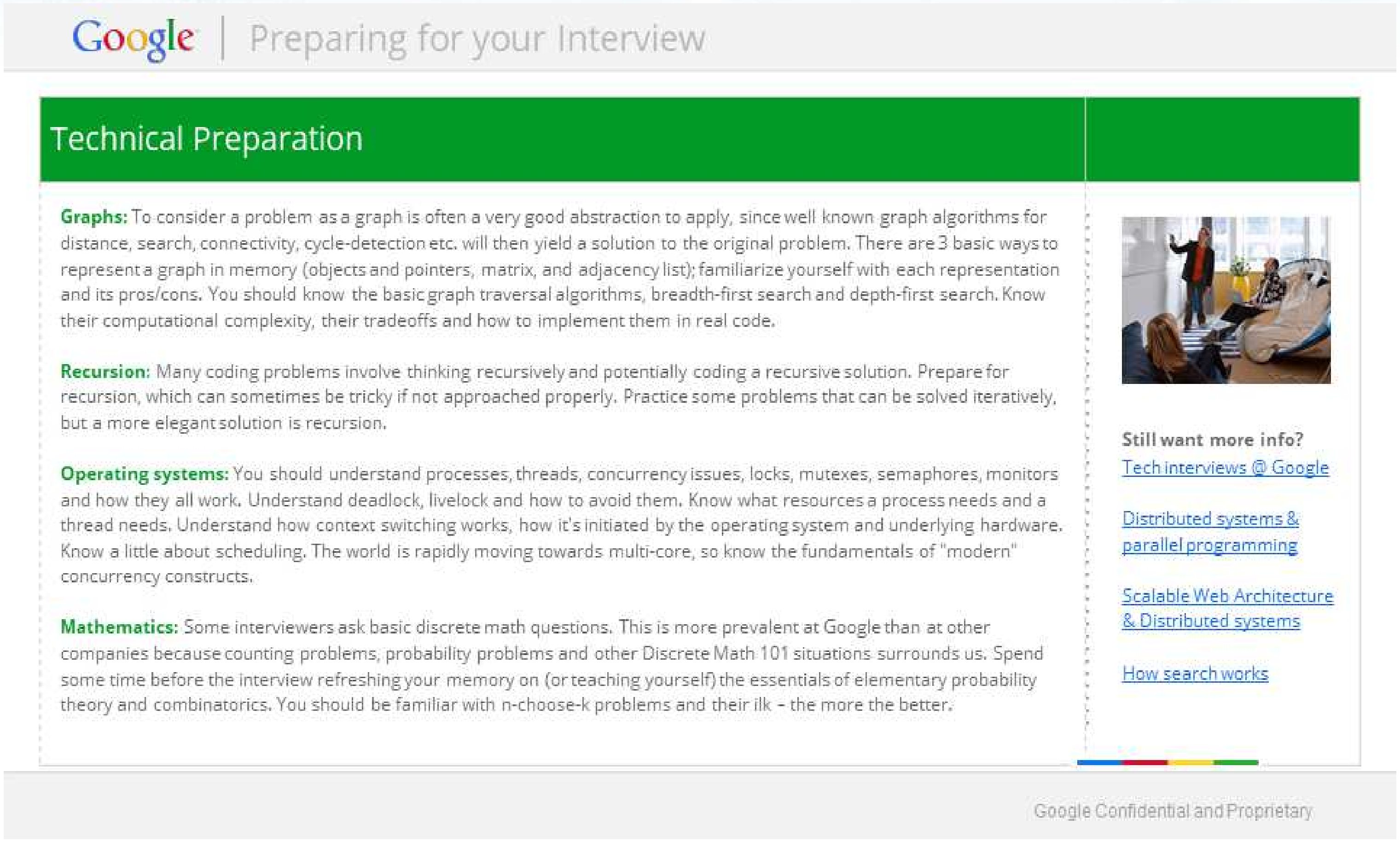
Trees: We recommend you know about basic tree construction, traversal and manipulation algorithms. You should be familiar with binary trees, n-ary trees, and trie-trees at the very least. You should be familiar with at least one flavor of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree. You'll want to know how it's implemented. You should know about tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

Min/Max Heaps: Heaps are incredibly useful. Understand their application and O() characteristics. We probably won't ask you to implement one during an interview, but you should know when it makes sense to use one.

Reminder:
Engineering Practicum interviews will focus on coding, algorithms and data structures and content will be level appropriate. See slide 2.

Google Confidential and Proprietary

Contents



Google | Preparing for your Interview

Technical Preparation

Graphs: To consider a problem as a graph is often a very good abstraction to apply, since well known graph algorithms for distance, search, connectivity, cycle-detection etc. will then yield a solution to the original problem. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize yourself with each representation and its pros/cons. You should know the basic graph traversal algorithms, breadth-first search and depth-first search. Know their computational complexity, their tradeoffs and how to implement them in real code.

Recursion: Many coding problems involve thinking recursively and potentially coding a recursive solution. Prepare for recursion, which can sometimes be tricky if not approached properly. Practice some problems that can be solved iteratively, but a more elegant solution is recursion.

Operating systems: You should understand processes, threads, concurrency issues, locks, mutexes, semaphores, monitors and how they all work. Understand deadlock, livelock and how to avoid them. Know what resources a process needs and a thread needs. Understand how context switching works, how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, so know the fundamentals of "modern" concurrency constructs.

Mathematics: Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because counting problems, probability problems and other Discrete Math 101 situations surrounds us. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of elementary probability theory and combinatorics. You should be familiar with n-choose-k problems and their ilk – the more the better.

Still want more info?

Tech interviews @ Google

Distributed systems & parallel programming

Scalable Web Architecture & Distributed systems

How search works

Google Confidential and Proprietary

Delivery Modes

- There is no distinction between theory and practice lessons
  - Lectures include practice lessons
  - 5 blocks of 1.5 ore
  - ECE students will be told when the lesson is optional for them

Delivery Modes

- Lectures and practice are extended with 20 additional hours in laboratory
  - 2 lab teams
    - 2 blocks of 1.5 ore (1 for each lab team)
    - Team A: A – LA, Monday 10.00-11.30, laib 4
    - Team B: LB – Z, Monday 11.30-13.00, laib 4
  - Problem solving in C language
    - From specs to code through editing, compilation, debugging, and execution of programs
    - Windonw operating system
    - Codebock (or similar) API (Application Programming Interface)

Texts, Readings, Handouts

- Material
  - Personal student's page (Politecnico portal)
    - Video-recordings
    - Calendar, rules and deadlines
    - Exams bookings and exam results
  - Teacher's personal WEB page
    - **htttp://fmgroup.polito.it/quer/**
    - Material used during all lectures and practice
      - Teacher names, course program and rules
      - Overheads, book references, other material
      - Laboratory exercises and solutions
      - Theory examination texts
      - Etc.

Texts, Readings, Handouts

- Material directly used in class
  - Slides
    - Teacher’s personal web page
    - Mainly from Prof. Paolo Camurati
  - Printed material
    - S. Quer, "Advanced Programming and Problem-Solving Strategeis in C. Part II: Algorithms and Data Structures", CLUT, Sept. 2017
    - S. Quer, "Advanced Programming and Problem-Solving Strategeis in C. Part IV: Exam-Based Problems", CLUT, Sept. 2017
  - Laboratory specs and solutions

Texts, Readings, Handouts

- Other printed material
  - Theory part

    Easier to follow but it adopts pseudo-code (non C)

    - T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms", McGraw-Hill.

    Harder to follow but it adopts C code

    - R. Sedgewick, "Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching" and "Algorithms in C, Part 5: Graph Algorithms", Addison-Wesley Professional
  - Programming part
    - B. W. Kernighan, D. M. Ritchie, "The C Programming Language", Prentice Hall, second edition.
    - P. Deitel, H. Deitel, "C: how to program", Prentice Hall, eight edition.

Assessment and Grading Criteria

- The exam consists in
  - A written exam
    - A theory part
    - A programming part
    - Maximum duration 2h30
  - An off-line section to check the written programming part and to up-load a final (complete and working) program (based on the written programmng part)
  - An oral exam

Assessment and Grading Criteria

- Written part
  - No books, notes, transparencies, cell phones, etc., are allowed
  - It includes
    - A theory part
    - A programming part
  - The theory part
    - Includes exercises and questions on all theoretical aspects presented during the class
      - Manual application of standard algorithms on data streams and data structures
    - Maximum 12 points
    - Maximum duration 50 minutes

The theory part will (often) be different for ECE students

Assessment and Grading Criteria

- The programming part
  - Checks problem solving ability in C
  - Available in two alternative modes
    - A "standard" problem solving task (program) the emphasis being on problem-solving and design skills
    - A "simplified" set of 3 C exercises, with less emphasis on design and more on the ability to use advanced C features
  - Maximum
    - 18 points for the "standard" part
    - 12 points for the "simplified" one
  - Maximum duration 1h45

The programming part will (often) be the same for ECE students but possibly solvable with a simplified strategy

Assessment and Grading Criteria

- The off-line section consists in
  - Make a copy of the program at the end of the written part
  - Verify the quality and correctness of the program within 3 working
  - Upload on the **course webpage** a copy of the working program
    - Comments must show changes (with respect to the version handed-in for ranking)
  - In case the above material is not uploaded, the written exam will not be ranked otherwise a mark will be uploaded on the course webpage

Assessment and Grading Criteria

- Oral examination
  - Students whose mark in the written part is larger or equal to 15/30 are entitled to sit for the oral exam
  - The oral examination consists in questions on all topics of the course (theory topics and problem solving)
- The final mark integrates partial results (written part and oral exam) and is not a mere average or sum of those parts

For ECE students the oral examination wil include only their program part

Results so far …

- Results "cohort" by "cohort"
  - From 2012-2012 to 2016-2017
  - Evaluation date: 01.10.2017

E = Enrolled
P = Passed

Accademic Year	2011-2012			2012-2013			2013-2014		
	E	#P	%P	E	#P	%P	E	#P	%P
2011-2012	33	8	24						
2012-2013	14	1	3	58	12	21			
2013-2014	11	2	6	40	6	10	56	12	21
2014-2015	6	2	6	30	10	17	28	4	7
2015-2016	4	1	3	18	4	7	23	6	11
2016-2017	3	0	0	12	1	2	16	5	9
Total passed		**14**	**42**		**33**	**57**		**27**	**48**
Dropped-out		**16**	**49**		**14**	**24**		**18**	**32**
Still enrolled		**3**	**9**		**11**	**19**		**11**	**20**

Results so far ...

Accademic Year	2014-2015			2015-2016			2016-2017		
	E	#P	%P	E	#P	%P	E	#P	%P
2014-2015	67	22	33						
2015-2016	37	2	3	62	18	29			
2016-2017	31	4	6	36	6	10	61	27	44
Total passed		**28**	**42**		**24**	**39**		**27**	**44**
Dropped-out		**12**	**18**		**8**	**13**		**-**	**-**
Still enrolled		**27**	**40**		**30**	**48**		**34**	**56**

From 10 (≤2015-2016) to
12 credits (≥ 2016-2017)

Results so far …

- Results for all academic years
  - From 2011-2012 to 2016-2017
  - Evaluation date: 01.10.2017

!!!

Total Number of …	Total	[%]
… students enrolled	334	100
… exam taken	331	99
… students who never take the exam	**149**	45
… students who dropped-out	**67**	20
… passed on total (153/334)	153	46
… passed on who takes the exam (153/(334-149))	153	83
Average mark	**22**	

Not considered 2017-2018 !

To summarize

- During the course we will face both theory and practice problems

To summarize

- During the course we will face both theory and practice problems
  - Theory is when you know everything but nothing works

To summarize

- During the course we will face both theory and practice problems
  - Theory is when you know everything but nothing works
  - Practice is when everything works but no one knows why

To summarize

- During the course we will face both theory and practice problems
  - Theory is when you know everything but nothing works
  - Practice is when everything works but no one knows why
  - **In this class, theory and practice will be combined: Nothing will work and no one will know why**

(possibly) Albert Einstein, 1879-1955