UESTC 1005 - INTRODUCTORY PROGRAMMING

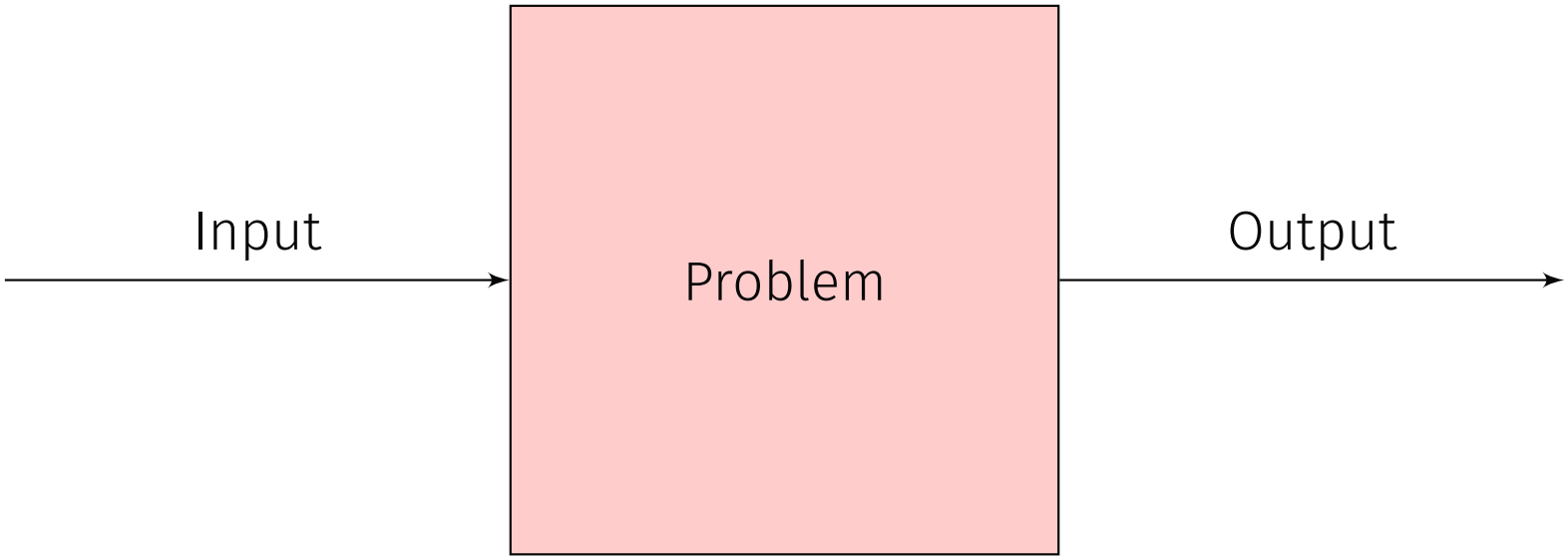
A Course on C Programming Language

Dr Hasan T Abbas
Hasan.Abbas@glasgow.ac.uk

Fall 2019

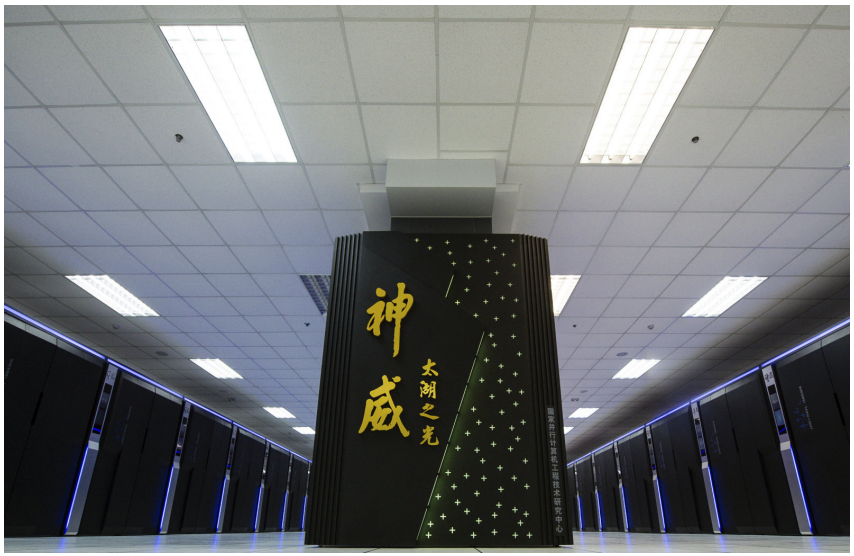
Glasgow College - UESTC

BACKGROUND



- Historically, we have found ways to solve our problems with the use of *numbers*.
- Computers are really good at representing/manipulating numbers.
- We need to tell the computer **exactly** what to do.
- Sequence of instructions is called a computer program

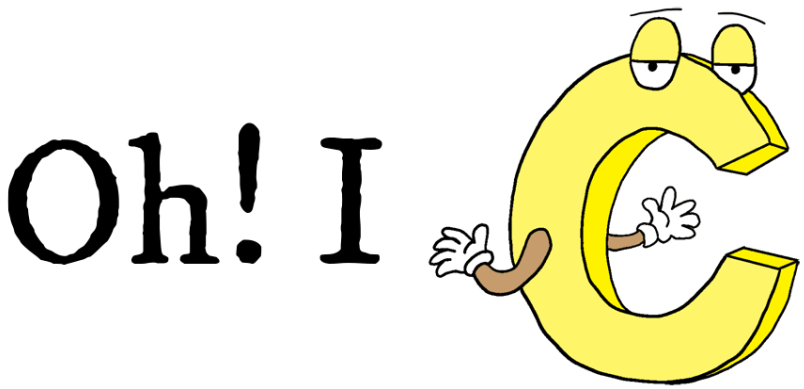
- Computers are phenomenally faster than us in performing calculations.
- Computers directly understand machine language only.
- Communication with computers require assemblers.
- High-level Languages require *compilers*



Shenwei Taihu zhi guang

```
grade = exams + quiz + labs;
```

- C is a by-product of UNIX operating system
- Developed in the 1970s at Bell Laboratories by Ken Thompson
- *High-level* Programming language
- Linux, Windows, and Android have been written in C



COURSE DESCRIPTION

AIM OF THIS COURSE

- To equip the candidate with the skills and knowledge necessary to write programs in the C programming language.
- To introduce the concepts necessary for the construction of larger programs.
- To foster the ability to adhere to specification when writing modules of larger programs.
- To develop skills in testing and debugging code.

INTENDED LEARNING OUTCOMES

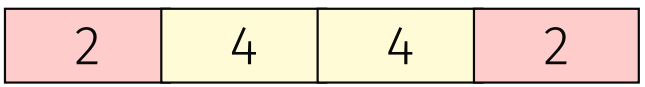
- By the end of the course, students will be able to:
  - write practical functioning C programming code which makes full use of the following constructs:
    - simple (int, float, char, string) and complex (arrays, structs, pointers) data types,
    - operators (algebraic, assignment, relational, Boolean), and functions;
    - flow control statements (do-while, while, for, if-else)
    - standard input and output library functions;
  - find and correct logical errors in student written code and example code.
  - design, write, compile and run simple programs using a commercial Integrated Development environment
  - Recognize correct syntax in C programs, and correct C programs with erroneous syntax
  - Recognize when such programs are running to a simple written specification.
  - Demonstrate the knowledge of technical English vocabulary

COURSE ASSESSMENT

- Written Final Exam 60 %
  - No books allowed
- Midterm 10 %
  - Tentative Week 8-9
- Lab Work 30 %
  - Total 8 labs, 16 exercise.
  - Attend all lab sessions to pass this course.
  - Show your results to the GTA in each lab session.
  - Submit lab reports on Moodle within a week.
  - If you miss any lab session, you will not receive any grade, i.e. “CW”.

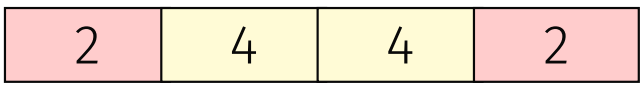
LAB GRADING SCHEME

- Each exercise is 10 points.
  - 0 marks for no-show
- Correct Code 4
  - Correct code which works without any error and produces the desired output. 4
  - Code works without any errors but produces an incorrect output. 3
  - Partially correct code, no or incorrect output. 2
  - Code doesn't compile, has syntax errors or doesn't work at all, and produces no output. 1

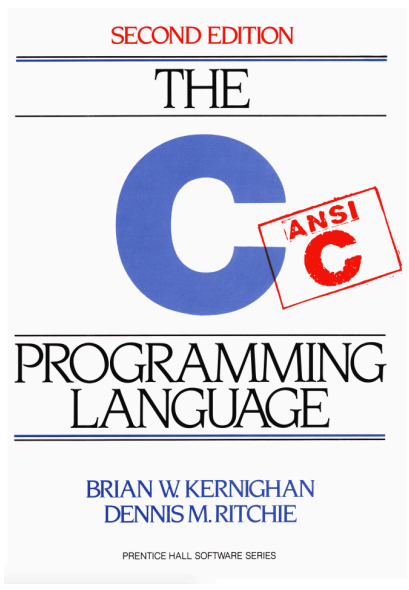


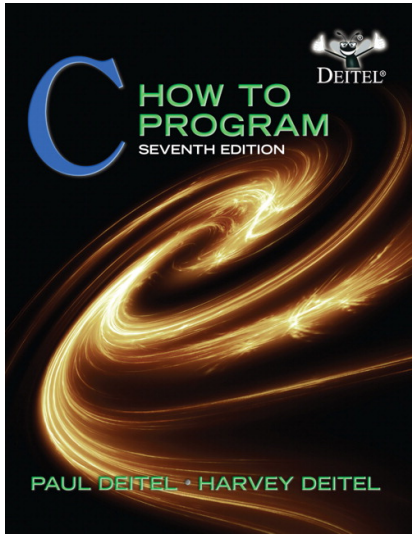
LAB GRADING SCHEME

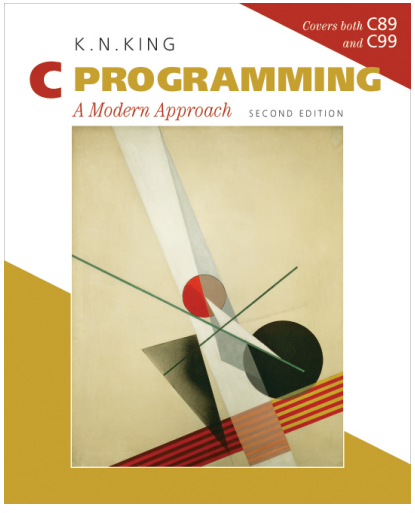
- Readability of the code 2
  - Proper indentation, documentation, understandable variable names, and comments
- Correct response to questions asked 4
  - Student demonstrating code works. 2
  - Correct response on any changes made to the code. 2



- Check Class page on Moodle
  - Lab Manuals, useful programming reference materials
- Textbook not required
  - Useful Reference Books







CLASS RULES

- Communication during the lectures and labs in English only.
- Use of cell phones not allowed during lectures
- No Cheating or copying during lab exercises, and exams
- Attendance mandatory for all lab sessions.
- In order to pass, students must attempt all the exams, complete and submit at least 12 lab exercises, i.e. 75% by weight of the total laboratory marks, to receive a grade.
- No grade or CW will be awarded if student fails to meet the above criteria
- The marks could be reduced according to the school policy, if you arrive late by more than 15 minutes at any lab session or submit your lab work and/or report late.

- Practice, practice and practice.
- Attend all lectures and lab sessions.
- Ask questions and seek for help!
- Study independently
- Go through the course material.

A SIMPLE C PROGRAM

```c
#include<stdio.h>
int main(){
   printf("\n \n I C a Python ;-) \n \n");
   return 0;
}
```

A VERY BAD C PROGRAM

University of Glasgow

```
v,i,j,k,l,s,a[99];
main() {
   for(scanf("%d",&s),
   *a-s,
   v=a[j*=v]-a[i],
   k=i<s,
   j+=(v=j<s&& (!k&&!!printf(2+"\n\n%c"-(!l<<!j),
     #Q" [1^v? (1^j) &1:2] ) && ++l || a [i] <s&&v&&v
     v|| (i==j?a[i+=k]=0:  ++a[i] )>=s*k&&++a[--i]);
}
```

READING REFERENCE MATERIAL

- Chapter 2
  - Deitel & Deitel - C-How to Program, 8^{th} ed.
- Chapters 1 & 2
  - King - C Programming - A Modern Approach, 2^{nd} ed.