1. General information about the exam

- The exam consists of a project, a written exam, and an oral exam.
- The project, which has to be developed by groups of two on a problem that changes at each exam session, consists of implementing an ANSI C program or library for that problem by following the methodology for developing software "in the small" presented during the course. The project has to be submitted between 17 and 10 days before the written exam (example: if the date of the written exam is Jan 29, the project must be submitted between Jan 12 at 00:01 and Jan 19 at 23:59). In case of early submission, the project is not considered; in case of late submission, a 3/30 penalty is applied for each day after the deadline. The project is passed if the mark is at least 18/30; the mark is valid, even if the written or oral exam is taken but not passed, until the third exam session after the one in which the project is submitted (this session is explicitly mentioned in the project evaluation form).
- Should the project be resubmitted in a subsequent exam call, the mark of the previously submitted project is canceled. Project resubmission in the same exam session can take place only once per session and in this case a 4/30 penalty is applied to the mark of the newly submitted project because the group can benefit from the correction of the previously submitted project (the correction of the new project in the same exam session may reveal errors that went undetected during the correction of the previously submitted project).
- The written exam, which can be individually taken only after passing the project and changes at each exam call, consists of 8 questions plus 2 exercises to carry out in 90 minutes. It is passed if the mark is at least 18/30; the mark is valid only for the exam call in which the written exam is taken.
- The oral exam, which can be individually taken only after passing the project and the written exam and is held on the same day as the written exam, consists of a discussion of the project and of the written exam plus further questions. If passed, it determines an adjustment between -5/30 and 5/30 of the average of the two previous marks, thus yielding the final mark.
- Students must register to the written exam through the student career digital management system; this registration is possible only in case of regular payment of tuition fees and is necessary for filling in the questionnaire on the quality of teaching. Moreover, at the written exam and at the oral exam students must bring a valid identification document (identity card, passport, driver license, academic record book, or academic card) and a black or blue pen.

1. Informazioni generali sull'esame

- *L'esame consiste in un progetto, una prova scritta e una prova orale.*
- *Il progetto, che deve essere sviluppato in gruppi da due su un problema che cambia ad ogni sessione d'esame, consiste nell'implementazione di un programma o di una libreria ANSI C per quel problema seguendo la metodologia di sviluppo software "in the small" presentata a lezione. Il progetto deve essere consegnato tra* 17 *e* 10 *giorni prima della prova scritta (esempio: se la data della prova scritta è il* 29/01*, il progetto deve essere consegnato tra le ore* 00:01 *del* 12/01 *e le ore* 23:59 *del* 19/01*). In caso di consegna anticipata, il progetto non viene considerato; in caso di consegna tardiva, viene applicata una penale di* 3/30 *per ogni giorno di ritardo. Il progetto è superato se il voto è di almeno* 18/30*; il voto rimane valido, anche se la prova scritta od orale viene sostenuta ma non superata, fino alla terza sessione d'esame successiva a quella in cui il progetto viene consegnato (tale sessione è esplicitamente indicata nella scheda di valutazione del progetto).*
- *Qualora il progetto venga riconsegnato in un successivo appello d'esame, il voto del progetto precedentemente consegnato viene annullato. La riconsegna del progetto nella medesima sessione d'esame è ammessa una sola volta per sessione e in tal caso al voto del nuovo progetto consegnato viene applicata una penale di* 4/30 *perché il gruppo può beneficiare della correzione del progetto precedentemente consegnato (la correzione del nuovo progetto nella medesima sessione d'esame potrebbe rivelare errori che erano passati inosservati durante la correzione del progetto precedentemente consegnato).*
- *La prova scritta, che può essere sostenuta individualmente solo dopo aver superato il progetto e cambia ad ogni appello d'esame, consiste in* 8 *domande più* 2 *esercizi da svolgere in* 90 *minuti. Essa è superata se il voto è di almeno* 18/30*; il voto rimane valido solo per l'appello d'esame in cui la prova scritta viene sostenuta.*
- *La prova orale, che può essere sostenuta individualmente solo dopo aver superato il progetto e la prova scritta e si svolge nello stesso giorno della prova scritta, consiste in una discussione del progetto e della prova scritta più ulteriori domande. Se superata, essa determina un aggiustamento compreso tra* -5/30 *e* 5/30 *della media dei due precedenti voti, producendo così il voto finale.*
- *È obbligatorio iscriversi alla prova scritta tramite il sistema di gestione digitale delle carriere studentesche; tale iscrizione è possibile solo in caso di regolare pagamento delle tasse universitarie ed è necessaria per compilare il questionario sulla qualità della didattica. Inoltre bisogna presentarsi alla prova scritta e alla prova orale con un documento di riconoscimento in corso di validità (carta d'identità, passaporto, patente di guida, libretto universitario o tessera universitaria) e una penna nera o blu.*

2. General information about the project

- The project must be sent to `marco.bernardo@uniurb.it` from the e-mail address belonging to the domain `@campus.uniurb.it` of either component of the group.
- The e-mail address belonging to the domain `@campus.uniurb.it` of the other component of the group must be in cc.
- The subject of the e-mail must be "consegna progetto programmazione procedurale".
- The body of the e-mail must contain for each component of the group on distinct rows: first name and last name, serial number, and year of enrollment (first, second, third, supplementary).
- The project must be an attachment in `.zip` or `.tar.gz` format comprising only:
  - A report in PDF format, called `relazione.pdf`, divided into sections according to the methodology for developing software "in the small" presented during the course.
  - All the `.c` and `.h` source files that have been developed to solve the assigned problem, each one with a name that is appropriate with respect to the assigned problem (hence different from generic names such as `exam.c`, `project.c`, `program.c`, `implementation.c`, `source.c`, `file.c`, `algorithm.c`, `functions.c|.h`, `library.c|.h`, `test.c`, `main.c`) as well as devoid of blanks.
  - A makefile, called `makefile` or `Makefile`, for compiling the source files.
- The project evaluation form will be sent via e-mail, at least 3 days before the written exam, to the address in the domain `@campus.uniurb.it` of each component of the group.
- The projects of different groups exhibiting basically the same software will all be nullified.
- Any violation of what established about the project submission, the report preparation, and the software development will determine a decrease of the project mark.
- Furthermore, each of the following serious violations will automatically determine the rejection of the project, which will not even be evaluated:
  - Submission of the project more than one day late.
  - Sending the project from an e-mail address not belonging to the specified domain.
  - Project submitted in a format different from the required one.
  - Failure to submit the report, some of the source files, or the makefile.
  - Report submitted in a format different from the required one.
  - Report without cover page or with sections not starting on a new page.
  - Absence of some of the sections in the report.
  - Absence of test numbering or execution of a number of tests less than the required one.
  - Impossibility to obtain the executable file through compilation.
- Warning: study recommended textbooks and lecture notes (with the latter not replacing the former) before starting the project, not after!

2. Informazioni generali sul progetto

- *Il progetto deve essere inviato a* `marco.bernardo@uniurb.it` *dall'indirizzo e-mail appartenente al dominio* `@campus.uniurb.it` *di una delle due persone che compongono il gruppo.*
- *L'indirizzo e-mail appartenente al dominio* `@campus.uniurb.it` *dell'altra persona che compone il gruppo deve comparire in cc.*
- *L'oggetto della e-mail deve essere "consegna progetto programmazione procedurale".*
- *Il corpo della e-mail deve contenere per ogni componente del gruppo su righe distinte: nome e cognome, numero di matricola e anno di corso (primo, secondo, terzo, fuori corso).*
- *Il progetto deve essere un allegato in formato* `.zip` *o* `.tar.gz` *comprendente esclusivamente:*
  - *Una relazione in formato PDF, denominata* `relazione.pdf`*, suddivisa in sezioni come da metodologia di sviluppo software "in the small" presentata a lezione.*
  - *Tutti i file sorgenti* `.c` *e* `.h` *che sono stati sviluppati per risolvere il problema assegnato, ciascuno con un nome che sia significativo rispetto al problema assegnato (quindi diverso da nomi generici come* `esame.c`*,* `progetto.c`*,* `programma.c`*,* `implementazione.c`*,* `sorgente.c`*,* `file.c`*,* `algoritmo.c`*,* `funzioni.c|.h`*,* `libreria.c|.h`*,* `test.c`*,* `main.c`*) nonché privo di spazi.*
  - *Un makefile, denominato* `makefile` *o* `Makefile`*, per compilare i file sorgenti.*
- *La scheda di valutazione del progetto verrà inviata tramite e-mail, almeno* 3 *giorni prima della prova scritta, all'indirizzo nel dominio* `@campus.uniurb.it` *di ogni componente del gruppo.*
- *I progetti di gruppi diversi che esibiscano praticamente il medesimo software verranno tutti annullati.*
- *Ogni inosservanza di quanto stabilito a proposito della consegna del progetto, della preparazione della relazione e dello sviluppo del software determinerà una riduzione del voto del progetto.*
- *Inoltre, ciascuna delle seguenti inosservanze gravi determinerà automaticamente l'insufficienza del progetto, che non verrà nemmeno valutato:*
  - *Consegna del progetto con più di un giorno di ritardo.*
  - *Invio del progetto da un indirizzo e-mail non appartenente al dominio specificato.*
  - *Progetto consegnato in un formato diverso da quello richiesto.*
  - *Mancata consegna della relazione, di qualcuno dei sorgenti o del makefile.*
  - *Relazione consegnata in un formato diverso da quello richiesto.*
  - *Relazione priva di pagina di copertina o con sezioni che non iniziano su una nuova pagina.*
  - *Assenza di qualcuna delle sezioni nella relazione.*
  - *Assenza della numerazione dei test o svolgimento di un numero di test inferiore a quello richiesto.*
  - *Impossibilità di ottenere l'eseguibile mediante compilazione.*
- *Nota bene: studiare i testi consigliati e le dispense (dove queste ultime non sostituiscono i libri) prima di iniziare il progetto, non dopo!*

3. Report preparation

- The report has to be written in correct Italian or English, using a font of size between 10 and 15. The text must be both left- and right-justified. For code and makefile, a monospace font like Courier is recommended to preserve alignments in the report, possibly of a reduced size with respect to the text so as to avoid that long lines wrap in the report thereby breaking indentation.
- The report has to start with a cover page showing at least the name of each component of the group and has to contain the following six sections each starting on a new page and featuring a sequence number:
  1. Problem Specification. Write faithfully the statement of the assigned problem.
  2. Problem Analysis. Organize it into the three subsections 2.1 Problem Input Data, 2.2 Problem Output Data, and 2.3 Intervening Relationships (to exploit in order to solve the problem), by abstracting from the algorithmic aspects that will be subsequently designed as well as from the implementation language that will be subsequently used.
  3. Algorithm Design. Organize it into the two subsections 3.1 Design Choices (to be illustrated and motivated with respect to data structures employed for the representation of input and output data, idea of the algorithmic solution based on the relationships between the aforementioned data, etc.) and 3.2 Steps of the Algorithm (to be listed with possible refinements and highlighting base/general cases for recursion), by abstracting from the characteristics of the language that will be subsequently used (we remind that input data validation is not a design choice, rather it is a requirement, and that the steps of the algorithm must not be written in ANSI C).
  4. Algorithm Implementation. Complete and translate the algorithm in ANSI C, then enclose in the report all the `.c` and `.h` files that have been developed, as well as the makefile, by paying attention to the preservation of indentation and alignments (if not required by the problem specification, the development of libraries must be adequately motivated by the provision of functionalities that are widely reusable).
  5. Program Testing. Conduct at least 10 meaningful tests of the complete execution of the program (we remind that compilation is part of neither execution nor testing), by faithfully showing for each test both the input data provided via keyboard/file and the corresponding results written onto screen/file, in addition to its sequence number.
  6. Program Verification. Organize it into the three subsections 6.1 Selected Code Fragment (group of at least three statements, hopefully not related to input/output, within one of the developed functions), 6.2 Property to Verify (logic formula formalizing the postcondition by referring only to identifiers occurring in the selected fragment and avoiding to repeat exactly the assignments contained in those statements), and 6.3 Verification (computation of the weakest precondition or application of the loop invariant theorem or induction principle in the case of repetitive or recursive statements, respectively, by avoiding shorthands for identifiers occurring in the selected statements).
- In the case that the problem specification requires the development of a library, in addition to showing input data, output data, intervening relationships, and algorithmic steps separately for each function to be exported, it is mandatory to write, enclose in the report, and submit through the attachment also a test program that includes the library and uses in its `main` all the functions indicated in the problem specification.

3. Preparazione della relazione

- *La relazione deve essere scritta in corretta lingua italiana o inglese, utilizzando un font di dimensione compresa tra* 10 *e* 15*. Il testo deve essere giustificato sia a sinistra che a destra. Per il codice e il makefile, un font monospazio come Courier è consigliato per mantenere gli allineamenti nella relazione, eventualmente di dimensione ridotta rispetto al testo così da evitare che linee lunghe vadano a capo nella relazione rovinando l'indentazione.*

- *La relazione deve iniziare con una pagina di copertina riportante almeno il nominativo di ogni componente del gruppo e deve contenere le seguenti sei sezioni ciascuna con inizio su una nuova pagina e numero progressivo:*

  1. *Specifica del Problema. Riportare fedelmente l'enunciato del problema assegnato.*
  2. *Analisi del Problema. Organizzarla nelle tre sottosezioni* 2.1 *Dati di Ingresso del Problema,* 2.2 *Dati di Uscita del Problema e* 2.3 *Relazioni Intercorrenti (da sfruttare ai fini della soluzione del problema), astraendo dagli aspetti algoritmici che verranno successivamente progettati così come dal linguaggio implementativo che verrà successivamente usato.*
  3. *Progettazione dell'Algoritmo. Organizzarla nelle due sottosezioni* 3.1 *Scelte di Progetto (da illustrare e motivare in riferimento a strutture dati utilizzate per la rappresentazione dei dati di ingresso e di uscita, idea della soluzione algoritmica basata sulle relazioni intercorrenti tra i dati anzidetti, ecc.) e* 3.2 *Passi dell'Algoritmo (da elencare con eventuali raffinamenti ed esplicitando casi base/generali per la ricorsione), astraendo dalle caratteristiche del linguaggio che verrà successivamente usato (si ricorda che la validazione dei dati di ingresso non è una scelta di progetto, ma una necessità, e che i passi dell'algoritmo non debbono essere scritti in ANSI C).*
  4. *Implementazione dell'Algoritmo. Completare e tradurre l'algoritmo in ANSI C, inserendo nella relazione tutti i file* `.c` *e* `.h` *sviluppati, come pure il makefile, facendo attenzione alla preservazione dell'indentazione e degli allineamenti (se non richiesto dalla specifica del problema, lo sviluppo di librerie deve essere adeguatamente motivato dalla messa a disposizione di funzionalità che siano ampiamente riutilizzabili).*
  5. *Testing del Programma. Effettuare almeno* 10 *test significativi di esecuzione completa del programma (si ricorda che la compilazione non fa parte né dell'esecuzione né del testing), riportando fedelmente per ciascun test sia i dati di ingresso introdotti da tastiera/file che i corrispondenti risultati prodotti su schermo/file, oltre al suo numero progressivo.*
  6. *Verifica del Programma. Organizzarla nelle tre sottosezioni* 6.1 *Brano di Codice Scelto (gruppo di almeno tre istruzioni, meglio se non di input/output, all'interno di una delle funzioni realizzate),* 6.2 *Proprietà da Verificare (formula logica che formalizza la postcondizione facendo riferimento solo a identificatori presenti nel brano scelto ed evitando di ripetere pari pari gli assegnamenti contenuti in quelle istruzioni) e* 6.3 *Svolgimento (calcolo della precondizione più debole o applicazione del teorema dell'invariante di ciclo o del principio di induzione in caso di istruzioni iterative o ricorsive, rispettivamente, evitando abbreviazioni degli identificatori presenti nelle istruzioni).*

- *Nel caso che la specifica del problema richieda lo sviluppo di una libreria, oltre a riportare dati di ingresso, dati di uscita, relazioni intercorrenti e passi algoritmici separatamente per ciascuna funzione da esportare, è obbligatorio scrivere, inserire nella relazione e consegnare tramite l'allegato anche un programma di test che include la libreria e usa nella sua* `main` *tutte le funzioni indicate nella specifica del problema.*

4. Software development

- The developed software (to be enclosed in the fourth section of the report) must be:
  - Readable:
    - Equipped with adequate comments that recall the phases of analysis and design.
    - Equipped with a comment before each definition of function.
    - Equipped with a comment next to each decl. of formal parameter (input, output, input/output).
    - Equipped with a comment next to each decl. of local variable (input, output, working).
    - Free of design choices not described/motivated in the third section of the report.
    - Free of global variables.
    - Free of programmer-introduced identifiers that do not recall what they represent.
    - Free of programmer-introduced identifiers in different styles (no mix of _/- or It/En).
    - Well indented and aligned (avoid long lines that wrap in the report).
    - Equipped with empty lines between declarations and statements as well as statement seqs.
    - Equipped with suitable blanks around binary operators within expressions.
  - Articulated in functions (avoid code redundancy):
    - All functions other than `main` must be declared before being defined.
    - The functions must be defined in invocation order starting from `main` (top down).
    - Function `main` must be neither too long, nor too short (a single invocation).
    - In the case of a library, the implementation file (`.c`) must not include the header file (`.h`) and all redeclarations contained in the latter must be preceded by `extern` (the two files must have the same name up to suffix).
  - Consistent with the principles of structured programming:
    - Free of `goto` statements.
    - Free of `exit` statements.
    - Free of `continue` statements.
    - Free of `break` statements not occurring at the end of a `case` of a `switch`.
    - Free of multiple `return` statements occurring in the body of the same function.
  - Portable (avoid references to non-standard libraries or specific character encodings).
  - Efficient (use data structures and algorithms that avoid waste of time and space).
  - Compilable through `gcc -ansi -Wall -O` without error messages and warnings.
  - Working correctly with respect to the assigned problem, without any limitation overcomable via dynamic memory allocation, and equipped with tight validation of every input.
- The developed makefile (to be enclosed in the fourth section of the report as well) must:
  - Generate an executable file having a name consistent with the one of the primary source (if the latter is `convert_distance.c`, the former will have to be `convert_distance`). In the case of a library, the executable file refers to the test program.
  - Have a name for the primary compilation target equal to the one of the executable file (if the latter is `convert_distance`, the former will have to be `convert_distance` too).
  - Have the makefile name in the dependency list of each target.
  - Use options `-ansi -Wall -O` in every occurrence of `gcc`.

4. Sviluppo del software

- *Il software sviluppato (da inserire nella quarta sezione della relazione) deve essere:*
  - *Leggibile:*
    - *Dotato di opportuni commenti che richiamano le fasi di analisi e progettazione.*
    - *Dotato di un commento prima di ogni definizione di funzione.*
    - *Dotato di un commento accanto a ogni dich. di parametro formale (input, output, input/output).*
    - *Dotato di un commento accanto a ogni dich. di variabile locale (input, output, lavoro).*
    - *Privo di scelte di progetto non descritte/motivate nella terza sezione della relazione.*
    - *Privo di variabili globali.*
    - *Privo di identificatori introdotti da chi programma non evocativi di ciò che rappresentano.*
    - *Privo di identificatori introdotti da chi programma in stili diversi (no misto di _/- o it/en).*
    - *Ben indentato e allineato (evitare linee lunghe che vanno a capo nella relazione).*
    - *Dotato di linee vuote tra dichiarazioni e istruzioni nonché tra sequenze di istruzioni.*
    - *Dotato di appropriate spaziature attorno agli operatori binari nelle espressioni.*
  - *Articolato in funzioni (evitare ridondanze di codice):*
    - *Tutte le funzioni diverse da* `main` *devono essere dichiarate prima di essere definite.*
    - *Le funzioni devono essere definite in ordine di chiamata a partire da* `main` *(top down).*
    - *La funzione* `main` *non deve essere troppo lunga, né troppo corta (una singola chiamata).*
    - *Nel caso di una libreria, il file di implementazione (*`.c`*) non deve includere il file di intestazione (*`.h`*) e tutte le ridichiarazioni contenute in quest'ultimo devono essere precedute da* `extern` *(i due file devono avere lo stesso nome a meno del suffisso).*
  - *Coerente coi principi della programmazione strutturata:*
    - *Privo di istruzioni* `goto`*.*
    - *Privo di istruzioni* `exit`*.*
    - *Privo di istruzioni* `continue`*.*
    - *Privo di istruzioni* `break` *che non si trovano alla fine di un* `case` *di uno* `switch`*.*
    - *Privo di molteplici istruzioni* `return` *nel corpo della stessa funzione.*
  - *Portabile (evitare riferimenti a librerie non standard o specifiche codifiche dei caratteri).*
  - *Efficiente (usare strutture dati e algoritmi che evitano sprechi di tempo e di spazio).*
  - *Compilabile tramite* `gcc -ansi -Wall -O` *senza che vengano segnalati errori o warning.*
  - *Funzionante correttamente rispetto al problema assegnato, senza alcuna limitazione superabile con l'allocazione dinamica della memoria, nonché comprensivo della validazione stretta di ogni input.*
- *Il makefile sviluppato (da inserire anch'esso nella quarta sezione della relazione) deve:*
  - *Generare un file eseguibile che abbia un nome coerente con quello del sorgente primario (se quest'ultimo è* `converti_distanza.c`*, il primo dovrà essere* `converti_distanza`*). Nel caso di una libreria, il file eseguibile è riferito al programma di test.*
  - *Avere il nome dell'obiettivo primario di compilazione uguale a quello del file eseguibile (se quest'ultimo è* `converti_distanza`*, anche il primo dovrà essere* `converti_distanza`*).*
  - *Avere il nome del makefile nella lista delle dipendenze di ogni obiettivo.*
  - *Usare le opzioni* `-ansi -Wall -O` *in ogni occorrenza di* `gcc`*.*