

Name and Surname	
Matricola	
Course	
Poli@Home <input type="checkbox"/> 1(AAAA-BARB) <input type="checkbox"/> 2(BARC-BOT) <input type="checkbox"/> 3(BOU-CASA) <input type="checkbox"/> 4(CASB-CHZ) <input type="checkbox"/> 5(CIA-COND) <input type="checkbox"/> 6(CONE-DELR) <input type="checkbox"/> 7(DELS-FEQ) <input type="checkbox"/> 8(FER-GEQ) <input type="checkbox"/> 9(GER-JOZ) <input type="checkbox"/> 10(JPA-MALI) <input type="checkbox"/> 11(MALJ-MOD) <input type="checkbox"/> 12(MOE-PAK) <input type="checkbox"/> 13(PAL-PORS) <input type="checkbox"/> 14(PORT-ROQ) <input type="checkbox"/> 15(ROR-SIGN) <input type="checkbox"/> 16(SIGO-TRIO) <input type="checkbox"/> 17(TRIP-ZZZ) <input type="checkbox"/> 18(Automotive) <input type="checkbox"/> Solo Prog <input type="checkbox"/> English <input type="checkbox"/>	

Theory

Question 1

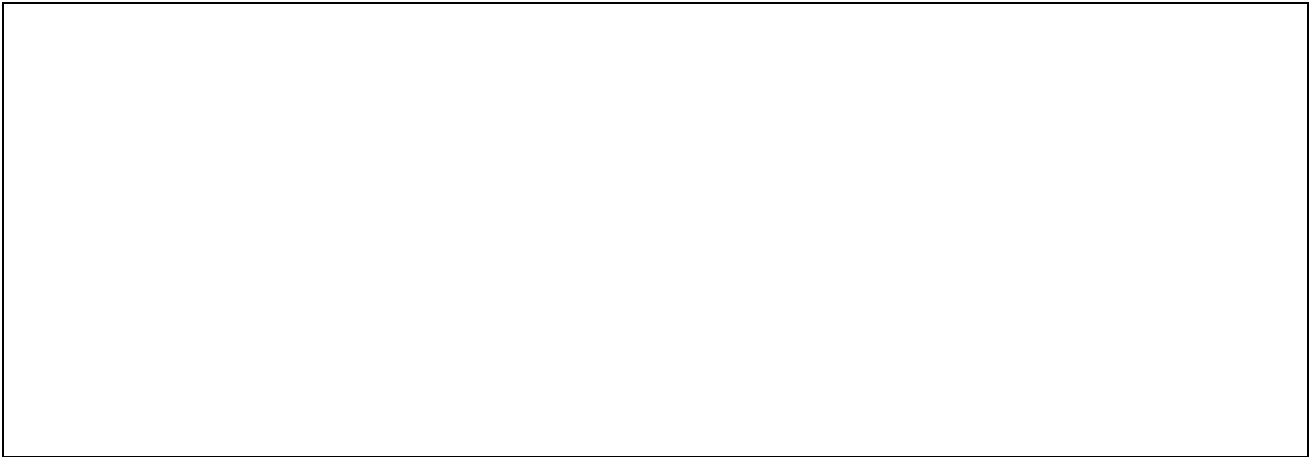
	<i>Result</i>
<p>Given the integer numbers $n1 = -7171$ e $n2 = -ABCD$, expressed in <u>base 16</u>, represent them in 2's complement on 16 bits, then calculate the sum S (operand in 2's complement) and discuss the potential presence of overflow.</p>	

Question 2

<p>If you want to store on a pen drive</p> <ul style="list-style-type: none"> A text of 3000 characters, on 50 lines, written in ASCII An audio file without compression, with a duration of 5 seconds and sampled at 30 Hz (30 samples per second), where each sample occupies 6 bytes <p>Calculate the total amount of bytes that are necessary, justifying the result.</p>

Question 3

Explain the meaning of logic gate and make at least two examples.
--

A large, empty rectangular box with a thin black border, occupying the upper half of the page. It is intended for the student to write their answers to the exam questions.

Programming

The file **map.txt** contains a map of size $N \times N$. The map represents the altitude in meters of a region. Possible values are in the range from 0 to 3000. N is a constant known a priori and defined by means of a **define** directive. Each line of the file describes a row of the map, and contains N integers separated by a space:

```
<h1,1> <h1,2> <h1,3> ... <h1,N>
<h2,1> <h2,2> <h2,3> ... <h2,N>
...
<hN,1> <hN,2> <hN,3> ... <hN,N>
```

Write a C program to analyze portions of the map. The program should receive 4 integers as arguments from the command line, representing the extremes of a rectangle (region of interest) within the map:

$$\langle x1 \rangle \langle y1 \rangle \langle x2 \rangle \langle y2 \rangle$$

where $(\langle x1 \rangle, \langle y1 \rangle)$ are the coordinates of the upper left corner of the rectangle, while $(\langle x2 \rangle, \langle y2 \rangle)$ are the coordinates of the lower right corner. The box $(0, 0)$ is at the upper left corner of the map, therefore $x1 \leq x2$ and $y1 \leq y2$. The program should provide the following information as output:

- Maximum altitude within the region of interest
- Minimum altitude (> 0) within the region of interest
- Percentage of sea within the region of interest (altitude == 0)
- Percentage of plain areas within the region of interest ($0 < \text{altitude} < 200$)
- Percentage of hills within the region of interest ($200 \leq \text{altitude} \leq 600$)
- Percentage of mountains within the region of interest (altitude > 600)

All the percentages have to be represented with exactly 2 decimal digits.

In case of error in the format of the file (integers not in the range from 0 to 3000, or insufficient number of data in the file) the program should print the message “File format error” and terminate the execution.

For example, if N is equal to 4 and the file contains the following data:

```
210 630 22 102
240 500 35 923
0 611 152 883
14 57 64 613
```

If the program is executed with arguments **0 0 1 2** the output should be:

```
Maximum altitude: 630
Minimum altitude: 210
Sea: 16.67%
Plain: 0.00%
Hills: 50.00%
Mountains: 33.33%
```

#include <stdio.h>

FILE *fopen(char *filename, char * mode) – Apertura di un file (mode: “r” lettura – “w” scrittura – “a” append)

FILE *freopen(char *filename, char * mode, FILE *file_pointer) - Riassegna un file puntatore ad un file diverso.

int fclose(FILE *file_pointer) - Chiude un file

int feof(FILE *file_pointer) - Controlla se e' stato incontrato un end-of-file in un file.

int fflush(FILE *file_pointer) - Svuota il buffer di un file.

int getchar(void) - Legge un carattere da "stdin" (tastiera)

int fgetc(FILE *file_pointer) - Prende un carattere da un file

char *gets(char *buffer) - Legge una riga da "stdin" (tastiera)

char *fgets(char *string, int maxchar, FILE *file_pointer) - Legge una riga da un file.

int printf(char *format_string, ...) - Scrive output formattato su "stdout" (schermo)

int fprintf(FILE *file_pointer, char *format_string, ...) - Scrive output formattato in un file.

int sprintf(char *string, char *format_string, ...) - Scrive output formattato su una stringa

int fputc(int c, FILE *file_pointer) - Scrive un carattere in un file

int putchar(int c) - Scrive un carattere su "stdout" (schermo)

int puts(char *string) - Scrive una stringa su "stdout" (schermo)

int fputs(char *string, FILE *file_pointer) - Scrive una stringa in un file.

int scanf(char *format_string, args) - Legge input formattato da "stdin" (tastiera)

int fscanf(FILE *file_pointer, char *format_string, args) - Legge input formattato da file

int sscanf(char *buffer, char *format_string, args) - Legge input formattato da una stringa

EOF – end of file (costante a valore negativo)

NULL - puntatore nullo (valore 0)

#include <stdlib.h>

double atof(char *string) - Converte una stringa in un valore in floating point.

int atoi(char *string) - Converte una stringa in un valore integer.

int atol(char *string) - Converte una stringa in un valore long integer.

void exit(int val) – Termina il programma, restituendo il valore ‘val’.

EXIT_FAILURE - costante per segnalare terminazione senza successo del programma con exit(); valore diverso da zero

EXIT_SUCCESS - segnala terminazione con successo del programma con exit(); vale 0

#include <string.h>

char *strcpy(char *dest, char *src) - Copia una stringa in un'altra. Restituisce dest

char *strncpy(char *s1, char *s2, size_t n) - Copia i primi "n" caratteri di s2 in s1.

Restituisce s1

int strcmp(char *s1, char *s2) - Confronta s1 e s2 per determinare l'ordine alfabetico (<0, s1 prima di s2, 0 uguali, >0 s1 dopo s2)

int strncmp(char *s1, char *s2, size_t n) - Confronta i primi "n" caratteri di due stringhe.

char *strncpy(char *s1, char *s2) - Copia s2 in s1. Restituisce s1

int strlen(char *string) - Determina la lunghezza di una stringa.

char *strcat(char *s1, char *s2, size_t n) - Aggiunge s2 a s1. Ritorna s1

char *strncat(char *s1, char *s2, size_t n) - Aggiunge "n" caratteri di s2 a s1. Ritorna s1

char *strchr(char *string, int c) - Cerca la prima occorrenza del carattere ‘c’ in string; restituisce un puntatore alla prima

occorrenza di c in s, NULL se non presente

char *strrchr(char *string, int c) - Cerca l'ultima occorrenza del carattere ‘c’ in string

char* strstr(char* s, char* t) - Restituisce un puntatore alla prima occorrenza di t

all'interno di s. Restituisce NULL se t non è presente in s.

char* strtok(char* s, const char* t) -

scomponi s in token, i caratteri che delimitano i token sono contenuti in t.

Restituisce il puntatore al token (NULL se non ne trova nessuno). Alla prima chiamata in s va inserita la stringa da scomporre e in t i caratteri che delimitano i vari token. Per operare sulla stessa stringa, alle successive chiamate al posto di s si deve passare NULL

#include <ctype.h>

int isalnum(int c) - Vero se ‘c’ e' alfanumerico.

int isalpha(int c) - Vero se ‘c’ e' una lettera dell'alfabeto.

int iscntrl(int c) - Vero se ‘c’ e' un carattere di controllo.

int isdigit(int c) - Vero se ‘c’ e' un numero decimale.

int islower(int c) - Vero se ‘c’ e' una lettera minuscola.

int isprint(int c) - Vero se ‘c’ e' un carattere stampabile.

int ispunct(int c) - Vero se ‘c’ e' un carattere di punteggiatura.

int isspace(int c) - Vero se ‘c’ e' un carattere spazio.

int isupper(int c) - Vero se ‘c’ e' una lettera maiuscola.

tolower(int c) - Converte ‘c’ in minuscolo.

int toupper(int c) - Converte ‘c’ in maiuscolo.

#include <math.h>

int abs(int n) – valore assoluto intero

long labs(long n) – valore assoluto long

double fabs(double x) – valore assoluto di x

double acos(double x) - arcocoseno

double asin(double x) - arcseno

double atan(double x) - arcotangente

double atan2(double y, double x) –

arcotangente di y/x.

double ceil(double x) – intero superiore a x

double floor(double x) – intero inferiore a x.

double cos(double x) – x in radianti

double sin(double x) – x in radianti

double tan(double x) – x in radianti

double cosh(double x) – coseno iperbolico

double sinh(double x) – seno iperbolico

double tanh(double x) – tangente iperbolica

double exp(double x) - e^x

double log(double x) - log(x).

double log10(double x) – logaritmo base 10

double pow(double x, double y) - x^y

int rand(void) – intero casuale tra 0 e RND_MAX.

int random(int max_num) – valore casuale tra 0 e max_num.

void srand(unsigned seed) – inizializza la sequenza di valori casuali

double sqrt(double x) – radice quadrata

#include <limits.h>

INT_MAX - Indica il più grande valore che è possibile rappresentare con un int.

INT_MIN - Indica il più piccolo valore che è possibile rappresentare con un int.

LONG_MAX - Indica il più grande valore che è possibile rappresentare con un long.

LONG_MIN - Indica il più piccolo valore che è possibile rappresentare con un long.

#include <float.h>

FLT_MAX, DBL_MAX - Indica il più grande valore che è possibile rappresentare con un float (o double)

FLT_MIN, DBL_MIN - Indica il più piccolo valore che è possibile rappresentare con un float (o double)

