|  |  |
| --- | --- |
| **Nome e Cognome** |  |
| **Matricola** |  |
| **Corso**  1(AAAA-BAHN) ☐ 2 (BAHO – BOGO) ☐ 3 (BOGP – CAPA) ☐ 4 (CAPB – CAVA) ☐ 5 (CAVB – CRIN) ☐ 6 (CRIO – DILA) ☐ 7 (DLLB – FIOU) ☐ 8 (FIOV –GIAM) ☐ 9 (GIAN – IMPO) ☐ 10 (IMPP – LUCI) ☐ 11 (LUCJ –MAZZ) ☐ 12 (MBAA – MUSS) ☐ 13 (MUST – PERE) ☐ 14 (PERF – RANI) ☐ 15 (RANJ – SANE) ☐ 16 (SANF –SPAT) ☐ 17 (SPAU – VACC) ☐ 18 (VACD –ZZZZ) ☐ Poli@Home☐ Es. (5 crediti)☐ | |

**Teoria**

**Domanda 1**

|  |  |
| --- | --- |
| Convertire i seguenti valori, rappresentati in modulo e segno, in complemento a 2   1. 10011 2. 00111 | *Risultato:*  *a.*  *b.* |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 2**

|  |  |
| --- | --- |
| Qual è l'errore assoluto massimo che si commette nelle seguenti rappresentazioni?   1. 0.101110 2. 10.1012 | *Risultato:*  *a.*  *b.* |
| *Passaggi più significativi per arrivare al risultato* | |

D**omanda 3**

|  |
| --- |
| Come può essere rappresentato in modo formale un algoritmo, oltre alle possibili rappresentazioni che si possono ottenere con i linguaggi di programmazione? |
|  |

**Programmazione**

Si realizzi un programma strutturato in linguaggio C che permetta di gestire le statistiche di una partita di calcio. Le informazioni sono contenute in un file il cui nome viene passato da linea di comando. Tale file contiene, uno per riga e separati da spazi: nome e cognome del giocatore, ruolo, chilometri percorsi, passaggi completati, tiri effettuati, goal fatti, falli fatti e falli subiti. Ad esempio, il file può contenere:

Mario Rossi difensore 7.2 75 0 0 15 5

Luca Verdi difensore 6.8 50 1 0 20 10

Andrea Bianchi centrocampista 8.3 90 5 1 7 5

Fabrizio Santana attaccante 6.0 25 5 1 4 20

Luigi Palladino portiere 0.5 25 0 0 0 3

Si facciano le seguenti assunzioni:

* Il numero di righe del file non può essere maggiore di 14
* I campi Nome e Cognome sono lunghi al più 30 caratteri e non possono contenere spazi
* Il ruolo può essere: portiere, difensore, centrocampista o attaccante
* I chilometri percorsi sono rappresentati come numeri reali con la precisione di una cifra decimale
* I restanti valori sono rappresentati come valori interi.

Il programma può ricevere da linea di comando uno dei seguenti parametri:

1. il flag "-a" e una stringa di caratteri che indica il cognome di un giocatore (si supponga non ci siano omonimie); in questo caso, il programma deve restituire a video tutte le statistiche relative al giocatore specificato.
2. il flag “-b” e una stringa di caratteri che indica un ruolo; in questo caso, il programma deve restituire le statistiche di tutti i calciatori che giocano in quel ruolo.
3. il flag "-c"; in questo caso, il programma deve visualizzare l'indicazione di **tutti** i giocatori che hanno fatto il massimo numero di goal.

Esempio, se il file dell'esempio precedente, contenente i risultati della gara, si chiama statistiche.txt:

C:\> prog.exe statistiche.txt –a Verdi

l'output del programma dovrà essere il seguente:

Luca Verdi: km 6.8, passaggi 50, tiri 1, goal 0, falli fatti 20, falli subiti 10

oppure:

C:\> prog.exe statistiche.txt –b difensore

l'output del programma dovrà essere il seguente:

Mario Rossi: km 7.2, passaggi 75, tiri 0, goal 0, falli fatti 15, falli subiti 5

Luca Verdi: km 6.8, passaggi 50, tiri 1, goal 0, falli fatti 20, falli subiti 10

oppure:

C:\> prog.exe statistiche.txt –c

l'output del programma dovrà essere il seguente:

Andrea Bianchi centrocampista 1 goal

Fabrizio Santana attaccante 1 goal

**#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 \*stpcpy (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 \*strcpy(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) -** scompone 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) -** arcoseno

**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) -** ex

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

**double log10 (double x ) –** logaritmo base 10

**double pow (double x, double y) -** xy

**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)