|  |  |
| --- | --- |
| **Nome e Cognome** |  |
| **Matricola** |  |
| **Corso**  1(AAAA - BARA) ☐ 2 (BARB – BOTS) ☐ 3 (BOTT – CAR) ☐ 4 (CAS – CORD) ☐ 5 (CORE – DIF) ☐ 6 (DIG – FIOR) ☐ 7 (FIOS - GIORD) ☐ 8 (GIORE – LANE) ☐ 9 (LANF – MARA) ☐ 10 (MARB – MOH) ☐ 11 (MOI – PAK) ☐ 12 (PAL – POLH) ☐ 13 (POLI – ROSA) ☐ 14 (ROSB – SIL) ☐ 15 (SIM – TR) ☐ 16 (TS – ZZ) ☐ E1 (AA – LZ) ☐ E2 (MA – ZZ) ☐ Poli@Home ☐ Es. (5 crediti)☐ | |

**Teoria**

**Domanda 1**

|  |  |
| --- | --- |
| Convertire i seguenti numeri:  23 **da** base 10 **a** base 2, binario puro su 8 bit  100011 **da** base 2, ca2 su 6 bit **a** base 10  0110100 **da** base 2 **a** base 16 | *Risposta* |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 2**

|  |  |
| --- | --- |
| Date le seguenti due funzioni booleane f e g, determinare se sono equivalenti costruendone le rispettive tabelle di verità: | Risposta |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 3**

|  |
| --- |
| Descrivere brevemente la struttura interna di una CPU |
|  |

**Programmazione**

Un file di testo contiene le informazioni relative alle presenze in partite ufficiali dei calciatori di serie A. Si vuole scrivere un programma in linguaggio C che permetta di estrarre delle statistiche su un campione di dati filtrati secondo specifici parametri.

Il nome del file contenente le informazioni sui calciatori è passato come primo   
argomento dalla linea di comando. Ogni riga di tale file si riferisce ad un singolo   
calciatore ed è formattata come segue:

<COGNOME> <NOME> <#PRESENZE> <MEDIA-VOTO> <SQUADRA>

dove COGNOME e NOME sono due campi distinti lunghi al più 40 caratteri ciascuno (senza spazi), #PRESENZE il numero di volte che il calciatore è sceso in campo durante una gara ufficiale, MEDIA-VOTO la media dei voti assegnati al calciatore durante le sue presenze, squadra il nome della squadra (lungo al più 25 caratteri) in cui attualmente milita il giocatore. Si facciano inoltre le seguenti assunzioni:

* Tutti i campi sono separati tra loro da un singolo spazio
* Il numero totale di calciatori non è noto a priori
* Non esistono omonimi
* Il contenuto del file è corretto

Le regole per il filtraggio dei dati sono contenute in un secondo file di nome filter.txt. In tale file ciascuna riga elenca il giocatore che deve essere escluso:

<SQUADRA> <#PRESENZE>

Il massimo numero di regole presenti nel file è 100.

A seguito del filtraggio, il programma dovrà stampare a video:

* l’elenco dei calciatori (con relative informazioni) che in una delle squadra specificate hanno fatto un numero di presenze superiore al numero specificato.
* il calciatore con il minor numero di presenze (fra quelli stampati)
* il calciatore con il maggior numero di presenze (fra quelli stampati)

In caso di ambiguità stampare il primo giocatore trovato.

Esempio:

seriA.txt filter.txt

|  |  |  |
| --- | --- | --- |
| Chiellini Giorgio 24 5.32 JUVENTUS  Consigli Andrea 35 6.13 ATALANTA  Marek Hamsik 38 6.45 NAPOLI  Diamanti Alessandro 10 6.88 NAPOLI  Vucinic Mirko 5 4.3 ATALANTA  Basta Dusan 28 6.25 UDINESE  Perrotta Simone 14 5.7 ROMA  Giovinco Sebastiano 3 4.5 UDINESE |  | ATLANTA 10  MILAN 15  UDINESE 25  NAPOLI 20  INTER 3 |
|  | | |
| C:\> **prog.exe serieA.txt**  Consigli Andrea 35 6.13 ATALANTA  Marek Hamsik 38 6.45 NAPOLI  Diamanti Alessandro 34 6.88 BOLOGNA  Basta Dusan 28 6.25 UDINESE  Presenze min.: 28 Basta Dusan  Presenze max.: 38 Marek Hamsik | | |

**#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