|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **COGNOME:** |  | **NOME** |  | **A3** |
| **MATRICOLA:** |  | | |
| **DOCENTE:** |  | | |

|  |  |
| --- | --- |
| **Domanda 1** | Risultato |
| Dati i seguenti numeri bimari espressi su 8 bit nel loro formato esadecimale  A=A1,   B=22  calcolare la loro somma considerando i numeri in binario puro e complemento a due indicando la presenza di overflow. | A2+B2 =\_\_\_\_\_\_\_\_\_\_\_ overflow:\_\_\_\_  ACA2+BCA2=\_\_\_\_\_\_\_\_\_\_\_ overflow:\_\_\_\_ |
| *Passaggi più significativi per arrivare al risultato* | |

|  |  |
| --- | --- |
| **Domanda 2** | Risultato |
| Data la seguente dichiarazione in C:  typedef struct book {    char title[50];    int chapter\_len[50];    float cost;    int book\_id;  } Book;  Book b[20];  Indicare la massima memoria richiesta per allocare la struttura. |  |
|

|  |
| --- |
| **Domanda 3** |
| Quale è la funzione dell’address bus, data bus e control bus in un calcolatore? |
|  |

|  |
| --- |
| **Domanda 4 (PROGRAMMAZIONE)** |

Scrivere un programma C che permetta di rilevare accessi sospetti a un server analizzando un file di log degli accessi chiamato “accessi.txt”.

Il file memorizza su ogni riga il tentativo di accesso di un utente al server nel seguente formato:

<DATA> <ORA> <Username> <STATUS>

Si facciano le seguenti ipotesi:

* La data è espressa nel formato AAAA-MM-GG.
* L’ora è espressa nel formato HH:MM:SS.
* Lo username è costituito da un massimo di 10 caratteri privi di spazi.
* Lo stato può assumere 2 valori: S per indicare un accesso corretto o F per indicare un accesso fallito.
* Il file è ordinato per data e ora crescenti e il suo contenuto può essere considerato corretto.
* **Non è noto il numero di righe che compongono il file.**

Il programma riceve sulla linea di comando lo username di 3 utenti da monitorare. Si definisce **ANOMALIA** due accessi consecutivi falliti da parte di un utente nello stesso giorno a distanza di meno di un minuto.

Il programma deve:

* Stampare le anomalie rilevate per gli utenti ricevuti in input (nell’ordine in cui vengono rilevate nel file accessi.txt) indicando username, data, numero di secondi tra due accessi falliti.
* Stampare l’intervallo medio tra due accessi falliti per tutte le anomalie rilevate (con una precisione di due cifre decimali)
* Identificare l’utente tra quelli ricevuti in input con il minor numero di anomalie rilevate

**Esempio**

Sia dato il seguente file accessi.txt

2015-01-01 12:00:01 rossi S

2015-01-02 12:00:01 bianchi F

2015-01-02 12:00:20 bianchi F

2015-01-03 00:00:01 rossi S

2015-01-03 11:12:20 verdi F

2015-01-03 11:12:55 brown S

2015-01-03 11:13:40 verdi F

2015-01-03 11:13:42 verdi F

2015-01-03 11:13:45 verdi S

2015-01-05 09:00:01 verdi F

2015-01-05 09:00:03 brown S

2015-01-05 09:00:05 verdi F

Eseguendo il programma con i seguenti parametri:

C:\> esame rossi bianchi verdi

Anomalie rilevate:

bianchi 2015-01-02 19

verdi 2015-01-03 02

verdi 2015-01-05 04

Intervallo medio tra due accessi falliti: 25/3 = 8.33

Utente con il minor numero di anomalie: rossi

**#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)