|  |  |
| --- | --- |
| **Nome e Cognome** |  |
| **Matricola** |  |
| **Corso**  Poli@Home☐ 1(AAAA-BARB)☐ 2(BARC-BOT)☐ 3(BOU-CASA)☐ 4(CASB-CHZ)☐ 5(CIA-COND)☐ 6(CONE-DELR)☐ 7(DELS-FEQ)☐ 8(FER-GEQ)☐ 9(GER-JOZ)☐ 10(JPA-MALI)☐ 11(MALJ-MOD)☐ 12(MOE-PAK)☐ 13(PAL-PORS)☐ 14(PORT-ROQ)☐ 15(ROR-SIGN)☐ 16(SIGO-TRIO)☐ 17(TRIP-ZZZ)☐ 18(Automotive)☐ Solo Prog☐ | |

**Teoria**

**Domanda 1**

|  |  |
| --- | --- |
|  | *Risultato* |
| Dato il seguente numero su 5 bit:  11101  scriverne il valore decimale interpretandolo come:   * Binario puro * Modulo e Segno * Complemento a 2 | Binario:  MS:  CA2: |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 2**

|  |  |
| --- | --- |
| Quanta memoria (in Byte) è necessaria per memorizzare un’immagine 320 x 200 pixel, 256 colori (formato bitmap non compresso)? |  |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 3**

|  |  |
| --- | --- |
| Scrivere la tabella della verità della seguente funzione booleana (dopo averla eventualmente semplificata) | **A B C F**  **0 0 0**  **0 0 1**  **0 1 0**  **0 1 1**  **1 0 0**  **1 0 1**  **1 1 0**  **1 1 1** |
| *Passaggi più significativi per arrivare al risultato* | |

**Programmazione**

Si realizzi in C un programma per consultare un conto telefonico.

La base dati consiste in un file di testo chiamato “conto.dat”, in cui ogni riga descrive una telefonata nel formato:

<numero\_telefonico> <data> <durata> <tipo>

Il numero telefonico è una stringa alfanumerica lunga al massimo 15 caratteri, la data è una stringa nel formato GG/MM/YYYY, la durata della telefonata è espressa in secondi come numero intero. Il tipo di telefonata è un carattere che può valere ‘E’ (telefonata in entrata) o ‘U’ (telefonata in uscita). I dati sono separati da uno spazio. Non si deve assumere nessun ordinamento per le righe, ed il numero di righe non è noto a priori.

Ad esempio:

3472222222 01/01/2011 120 E

3472222222 02/01/2011 321 E

3471111111 01/01/2011 38 E

3472222222 08/01/2011 100 U

3471111111 01/02/2011 38 E

3473333333 01/02/2011 100 U

3472222222 10/01/2011 100 U

Si desidera realizzare un programma per visualizzare alcune statistiche sui numeri chiamati. Il programma riceve come unico parametro sulla linea di comando un numero di telefono e deve visualizzare sullo schermo:

* I minuti totali di chiamate in uscita verso quel numero, con due cifre decimali
* La durata in minuti con due cifre decimali e la data della chiamata in uscita più lunga verso quel numero
* Il costo totale delle telefonate in uscita da quel numero, sapendo che la tariffazione è di 20 cent. di scatto alla risposta, più 12 cent. al minuto (con tariffazione al primo secondo di ogni minuto).

Si assuma che il formato del file sia corretto.

Ad esempio:

**C:\> PROG.EXE 3472222222**

Minuti totali di chiamate in uscita: 3,33 minuti

Chiamata in uscita più lunga: 1,66 minuti effettuata il 08/01/2011

Costo totale chiamate in uscita: € 0,88

**#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)

|  |  |
| --- | --- |
| **Nome e Cognome** |  |
| **Matricola** |  |
| **Corso**  Poli@Home☐ 1(AAAA-BARB)☐ 2(BARC-BOT)☐ 3(BOU-CASA)☐ 4(CASB-CHZ)☐ 5(CIA-COND)☐ 6(CONE-DELR)☐ 7(DELS-FEQ)☐ 8(FER-GEQ)☐ 9(GER-JOZ)☐ 10(JPA-MALI)☐ 11(MALJ-MOD)☐ 12(MOE-PAK)☐ 13(PAL-PORS)☐ 14(PORT-ROQ)☐ 15(ROR-SIGN)☐16(SIGO-TRIO)☐17(TRIP-ZZZ)☐18(Automotive)☐ Es. (5 crediti)☐ | |

**Teoria**

**Domanda 1**

|  |  |
| --- | --- |
|  | *Risultato* |
| Dato il seguente numero su 5 bit:  11011  scriverne il valore decimale interpretandolo come:   * Binario puro * Modulo e Segno * Complemento a 2 | Binario:  MS:  CA2: |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 2**

|  |  |
| --- | --- |
| Quanta memoria (in Byte) è necessaria per memorizzare un’immagine 320 x 400 pixel, 128 tonalità di grigio (formato bitmap non compresso) |  |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 3**

|  |  |
| --- | --- |
| Semplificare la seguente funzione e poi scriverne la tabella della verità | **A B C F**  **0 0 0**  **0 0 1**  **0 1 0**  **0 1 1**  **1 0 0**  **1 0 1**  **1 1 0**  **1 1 1** |
| *Passaggi più significativi per arrivare al risultato* | |

**Programmazione**

Si realizzi in C un programma per consultare un conto telefonico.

La base dati consiste in un file di testo chiamato “conto.dat”, in cui ogni riga descrive una telefonata nel formato:

<numero\_telefonico> <data> <durata> <tipo>

Il numero telefonico è una stringa alfanumerica lunga al massimo 15 caratteri, la data è una stringa nel formato GG/MM/YYYY, la durata della telefonata è espressa in secondi come numero intero. Il tipo di telefonata è un carattere che può valere ‘E’ (telefonata in entrata) o ‘U’ (telefonata in uscita). I dati sono separati da uno spazio. Non si deve assumere nessun ordinamento per le righe, ed il numero di righe non è noto a priori.

Ad esempio:

3472222222 01/01/2011 120 E

3472222222 02/01/2011 321 E

3471111111 01/01/2011 38 E

3472222222 08/01/2011 100 U

3471111111 01/02/2011 38 E

3473333333 01/02/2011 100 U

3472222222 10/01/2011 100 U

Si desidera realizzare un programma per visualizzare alcune statistiche sui numeri chiamati. Il programma riceve come unico parametro sulla linea di comando un numero di telefono e deve visualizzare sullo schermo:

* I minuti totali di chiamate in entrata verso quel numero, con due cifre decimali
* La durata in minuti con due cifre decimali e la data della chiamata in entrata più corta verso quel numero,
* Supponendo che l’utente abbia abilitato una autoricarica, calcolare la ricarica totale ottenuta sapendo che ogni telefonata dà diritto a un bonus di 15 cent. alla risposta, più 5 cent. al minuto (con tariffazione all’ultimo secondo di ogni minuto).

Si assuma che il formato del file sia corretto.

Ad esempio:

**C:\> PROG.EXE 3472222222**

Minuti totali di chiamate in entrata: 7,35 minuti

Chiamata in entrata più corta: 2,00 minuti effettuata il 01/01/2011

Valore totale ricarica: € 0,65

**#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)

# Correzione (Totale 33 punti – lode > 31)

## Teoria : 6 punti (2 per esercizio)

Soglia: 4/6

## Programmazione: 27 punti

### Difficoltà:

* lettura formattata del file
* conversione da secondi in minuti richiede l’uso di un float
* confronto tra stringhe, copia di stringhe (date)
* calcolo del costo/ricarica totale richiede di contare il numero di telefonate fatte (in E o U)
* necessità di distinguere per l’output il caso particolare di località non presente nel file

I punti sono da vedere “in negativo”. Ad esempio: sbagliare su argc/argv non può togliere più di 3 punti.

|  |  |
| --- | --- |
| Argc/argv | -3 |
| Utilizzo di fopen, fclose | -1 |
| Parsing dell’input | -3 |
| Conversione da secondi a minuti | -2 |
| Stringhe (confronto tra i numeri di telefono, copia date) | -4 |
| Calcolo totale/più lunga/più corta | -3 |
| Calcolo del costo/ricarica totale | -3 |
| Gestione casi particolari (numero non presente nel file) | -2 |
| Tentativo di memorizzare il file in una struttura dati | -6 |

Soglia: 14/24