|  |  |
| --- | --- |
| **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**

|  |  |
| --- | --- |
| Effettuare le seguenti operazioni in CA2. Indicare se l’operazione genera overflow e motivare la risposta.   1. 00011+01111 2. 10111+10011 | *Risultato* |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 2**

|  |  |
| --- | --- |
| Si dimostri, mediante la tavola della verità, se la seguente relazione è vera  (not A) or (A and B) = (A and (not B)) | Risposta |
| *Passaggi più significativi per arrivare al risultato* | |

**Domanda 3**

|  |
| --- |
| Descrivere brevemente le caratteristiche principali della memoria RAM. |
|  |

**Programmazione**

Si realizzi un programma in C per consultare un file di testo contenente le giacenze di magazzino di una panetteria. Ogni riga del file, riferita a un determinato mese, è caratterizzata dai seguenti campi separati da uno spazio l’uno dall’altro.

<tipo-alimento> <N1> <N2> <N3>

Il tipo-alimento è un carattere che può assumere quattro valori: ‘F’ (farina), ‘V’ (lievito), ‘Z’ (zucchero) e ‘L’ (latte), . N1, N2 e N3 sono 3 numeri reali che rappresentano le giacenze di magazzino di un tipo di alimento in un mese. Per la farina e lo zucchero l’unità di misura è il Kg (chilogrammo), per il lievito è il g (grammo), e per il latte è il l (litro). Si facciano le seguenti assunzioni:

* Il numero di righe del file non è noto a priori,
* il formato del file sia corretto,
* N1 presente sulla prima riga di ogni tipo-alimento è corretto,
* Il primo giorno del mese vengono acquistati e trasferiti in magazzino 150 Kg di farina, 40 kg di zucchero, 1200 g di lievito e 60 l di latte.

Un file di esempio è il seguente:

F 250.50 165.18 135.50

L 68.50 42.00 22.50

Z 52.00 24.50 17.00

V 1200.00 750.50 50.00

F 285.50 215.50 155.50

L 82.50 85.00 20.00

Z 57.00 32.00 12.00

V 950.00 650.00 250.00

Il programma riceve da linea di comando il nome del file contenente le giacenze di magazzino e deve:

* verificare, per ogni tipo di alimento, che i valori numerici presenti su ogni riga sia una sequenza decrescente. Se la sequenza è decrescente, il consumo reale del mese è pari alla differenza tra N1 e N3, altrimenti si consideri un consumo forfettario mensile pari a 150 Kg per la farina, 30 kg per lo zucchero, 1000 g per il lievito e 50 l per il latte;
* verificare, per ogni tipo di alimento, che N1 di una riga sia uguale alla somma di N3 della riga precedente associata allo stesso tipo di alimento e la relativa quantità acquistata (e.g., N1 della quinta riga, associata alla farina, è uguale alla somma di N3 della riga precedente associata alla farina e 150kg). Se la condizione non è verificata stampare un messaggio (vedi esempio).
* Per ogni tipo di alimento visualizzare sullo schermo le seguenti informazioni:
  + il consumo totale,
  + la spesa totale, considerando che il costo della farina è pari a 0.20 Euro/Kg, del lievito è 0.02 Euro/g, dello zucchero è 0.7 Euro/Kg, e del latte è 1.05 Euro/l,
  + la spesa media mensile (spesa totale/numero di mesi considerati).

Ad esempio:

**C:\> PROG-TDE13072012.EXE giacenze.txt**

Riga 8: V N1=950.00 diverso da N3\_precedente=50.00 + 1200

Tipo alimento Consumo Totale Spesa totale Spesa media mensile

Farina 245.00 Kg 49.00 Euro 24.50 Euro

Latte 96.00 l 100.80 Euro 50.40 Euro

Zucchero 80.00 Kg 56.00 Euro 28.00 Euro

Lievito 1850.00 g 37.00 Euro 18.50 Euro

**#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)