ESERCIZIO 1

- Scrivere una funzione che data una stringa A calcoli la sua lunghezza.

```
int lunghezza(char A[]);
```

- Scrivere una funzione che date due stringhe A e B copi il contenuto di A in B e restituisca il numero di caratteri copiati.

```
int copiastr(char A[], char B[]);
```

- Scrivere una funzione che date tre stringhe A, B e C concateni in C il contenuto di A e B e restituisca il numero di caratteri copiati in C

```
int conc(char A[],char B[],char C[]);
```

Esercizio 1 - Soluzione

```
int lunghezza(char A[]) {
   int result = 0;
   while (A[result] != '\0')
        result++;
   return result;
}

int copiastr(char src[], char dest[]) {
   int result = 0;
   while (src[result] != '\0') {
        dest[result] = src[result];
        result++;
   }
   dest[result] = '\0';
   return result;
}

int conc (char s1[], char s2[], char dest[]) {
   int result;
   result = copiastr(s1, dest);
   result = result + copiastr(s2, &dest[result]);
   return result;
}
```

ESERCIZIO 2

- Al fine di stampare degli indirizzi su delle buste, è necessario comporre la prima parte dell'indirizzo come "Cognome Nome" o "Cognome N."
- Si realizzi una funzione che riceva come parametri:
  - il cognome
  - il nome
  - una stringa che conterrà la prima parte dell'indirizzo
  - la lunghezza massima della stringa indirizzo

ESERCIZIO 2

- La funzione deve copiare/concatenare nell'indirizzo il cognome seguito dal nome, avendo cura di rispettare le dimensioni della stringa indirizzo. Qualora la stringa indirizzo sia troppo piccola per contenere entrambi, la funzione provi a comporre la stringa come "Cognome N."
- Qualora neanche ciò sia possibile, la funzione ritorni un codice di errore opportuno (esempio -1)
- Se non si verifica nessun errore la funzione deve restituire il numero di caratteri nella stringa Indirizzo

ESERCIZIO 2

- Si realizzi una funzione che riceva come parametri:
  – il cognome
  – il nome
  – una stringa che conterrà la prima parte dell'indirizzo
  – la lunghezza massima della stringa indirizzo

```
int indirizzo(char Cognome[], char Nome[],
  char Indirizzo[],int dim);
```

Esempio: Se il cognome è Rossi e il nome è Mario e la dimensione dim = 15 allora la stringa Indirizzo sara' "Rossi Mario". Se invece la dimensione fosse 8 allora la stringa indirizzo sarebbe "Rossi M.". Se la dimensione fosse 5 allora verrebbe restituito un codice di errore -1

ESERCIZIO 2

- Per la risoluzione di questo esercizio si utilizzino le funzioni realizzate nel primo esercizio.

- Si provi a organizzare il progetto su piu' file

- Una volta terminato l'esercizio lo si rifaccia utilizzando le funzioni di libreria <string.h>
  - strlen() per determinare la lunghezza di una stringa
  – strcat() o strcpy() per comporre in indirizzo la nuova stringa

Esercizio 2 - Soluzione

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>


#define TRUE    1
#define FALSE   0

#define RESULT_OK 1
#define RESULT_ADDRESS_TOO_LONG -1
#define RESULT_COMPRESSED_NAME -2

#define MAX 50

typedef int BOOL;
typedef int resultType;
```

Esercizio 2 - Soluzione

```
resultType componiIndirizzo(char * cognome, char * nome, char * indirizzo,
                                                          int maxChars) {
    int requiredChars;
    int compressNome = 0, size;

    strcpy(indirizzo, ""); //inizializzazione...
    requiredChars = strlen(cognome) + 1 + strlen(nome);
    if (requiredChars > maxChars) {
          requiredChars = strlen(cognome) + 3;
          if (requiredChars > maxChars)
                    return RESULT_ADDRESS_TOO_LONG;
          else      compressNome = 1;
    }
    strcat(indirizzo, cognome);
    strcat(indirizzo, " ");
    if (!compressNome)
          strcat(indirizzo, nome);
    else {
          size = strlen(indirizzo);
          indirizzo[size] = nome[0];
          indirizzo[size+1] = '.';
          indirizzo[size+2] = '\0';     }
    if (!compressNome)
          return RESULT_OK;
    else
          return RESULT_COMPRESSED_NAME;
}
```

Esercizio 2 - Soluzione

```
void handleError(resultType result) {
    switch (result) {
          case RESULT_OK:
                printf("Nessun errore occorso!\n"); break;
          case RESULT_ADDRESS_TOO_LONG:
                printf("L'indirizzo e' troppo lungo...\n"); break;
          case RESULT_COMPRESSED_NAME:
                printf("Il nome e' stato compresso...\n");       break;
          default:
                printf("Unknown Error!\n");
    }
}

int main(void)
{
    char indirizzo[MAX];
    resultType result;

    result = componiIndirizzo("Chesani", "Federico", indirizzo, MAX-1);
    if ((result == RESULT_OK) || (result == RESULT_COMPRESSED_NAME)) {
          printf("%s\n", indirizzo);
          printf("Lunghezza indirizzo: %d\n", strlen(indirizzo));
    }
    else
          handleError(result);
    system("PAUSE");
    return (0);
}
```

ESERCIZIO 3

- Scrivere una funzione C che, data una stringa A ed una stringa B, calcoli il numero di occorrenze della stringa A in B.
- `int occorrenze(char A[], char B[]);`
- Ad esempio, se B="tre tigri contro tre tigri" ed A="tr", deve restituire 3.

Esercizio 3 - Soluzione

```
int occorrenze(char a[], char b[]) {
   int result = 0;
   int trovato = 0;
   int i=0, j=0, temp;

   while (b[j] != '\0') {
        if (b[j] == a[i]) {
                trovato = 1;
                temp = j;
                while (a[i] != '\0' && trovato) {
                        if (a[i] == b[j]) {
                                i++;
                                j++;
                        }
                        else {
                                trovato = 0;
                        }
                }
                if (trovato)
                        result++;
                j = temp;
                i=0;
        }
        j++;
   }
   return result; }
```

Esercizio 4

Scrivere una procedura ricorsiva:

void printchar(char stringa[])

che stampi, *ricorsivamente*, tutti i caratteri contenuti in **stringa**, un carattere per linea, assumendo che **stringa** sia *ben formata*.

Esercizio 4 – Soluzione

```
void printchar (char stringa[]) {
  if (stringa[0] == '\0')
      return;
  else {
      printf("%c\n", stringa[0]);
      printchar(&(stringa[1]));
  }
}
```

ESERCIZIO 5

Si realizzi un programma C che legga da utente i dati relativi ad alcuni corsi. In particolare, per ogni corso vengono dati:

- ***denominazione*** *del corso: una stringa di 20 caratteri che riporta il nome del corso;*
- ***cognome*** *del docente: una stringa di 15 caratteri che rappresenta il cognome del docente del corso;*
- ***iscritti****: un intero che indica il numero di studenti che frequentano il corso.*

Il programma deve stampare la denominazione del corso e il cognome del docente relativi a tutti i corsi che hanno il numero di iscritti maggiore o uguale alla media aritmetica degli iscritti (calcolata su tutti i corsi).

ESERCIZIO 5

Attenzione: abbiamo bisogno di un ARRAY di strutture !!!!!
Esempio: l'utente inserisce i seguenti dati per 3 corsi

analisi
obrecht
55
fond.inf
milano
40
geometria
ferri
37

analisi	***fond.inf***	***geometria***
obrecht	***milano***	***ferri***
55	***40***	***37***

La media e' di 44 quindi il programma stampa:

analisi
obrecht

ESERCIZIO 5 - Soluzione

```
#include <stdio.h>
#define N 30
typedef struct stud {
            char denominazione[20];
            char cognome_docente[15];
            int studenti;
    } corso;

int main() {
    int i, nc;
    float somma media;
    corso corsi[N];
    printf("Inserisci il numero dei corsi ");
    scanf("%d", &nc);
    /* inserimento dati */
    for (i=0; i<nc; i++) {
            printf("Inserisci il nome del corso ");
            scanf("%s",corsi[i].denominazione);
            printf("Inserisci il cognome del docente ");
            scanf("%s",corsi[i].cognome_docente);
            printf("Inserisci il numero degli iscritti");
            scanf("%d",&corsi[i].studenti); }
```

Continua...

ESERCIZIO 5 - Soluzione

```
somma=0;
for (i=0; i< nc; i++)
    somma=somma + corsi[i].studenti;
media= somma/nc;
for (i=0; i< nc; i++)
    if (corsi[i].studenti>=media)
            printf("%s %s\n",corsi[i].denominazione,
                    corsi[i].cognome_docente);
}
```

ESERCIZIO 6

1) Si scriva un programma C che legga una serie di dati e li memorizzi primo vettore SQUADRE (di dimensione 3) contenente strutture (**`struct squadra`**) del tipo:

- ***nome squadra*** *(stringa di lunghezza 20)*
- ***codice squadra*** *(intero)*
- ***goal fatti*** *(intero)*
- ***goal subiti*** *(intero)*

2) Stampi a terminale tutti i nomi e codici delle squadre che hanno fatto un numero di goal maggiore del numero dei goal subiti.

3) Letto a terminale un codice di una squadra stampi a video il nome della squadra, i goal fatti e i goal subiti.

ESERCIZIO 6

Attenzione: abbiamo bisogno di un ARRAY di strutture !!!!!
Esempio: l'utente inserisce i seguenti dati per 3 squadre

juventus
1
10
12
milan
2
7
6
inter
3
13
11

juventus	*milan*	*inter*
1	*2*	*3*
10	*7*	*13*
12	*6*	*11*

2) Viene stampato a video
milan 2
inter 3
3) Se l'utente digita 1 viene stampato
juventus 10 12

ESERCIZIO 6 - Soluzione

```
#include <stdio.h>
#define N 30

typedef struct squadra{
        char nome[20];
        int codice;
        int goal_fatti, goal_subiti;
} SQUADRA;
void main() {
        int i, ns, cod, T;
        SQUADRA SQUADRE[N];

        printf("Inserisci il numero delle squadre");
        scanf("%d", &ns);

        /* inserimento dati */
        for (i=0; i<ns; i++) {
                printf("Inserisci nome, codice, goal fatti e subiti \n");
                scanf("%s",SQUADRE[i].nome);
                scanf("%d",&SQUADRE[i].codice);
                scanf("%d",&SQUADRE[i].goal_fatti);
                scanf("%d",&SQUADRE[i].goal_subiti);
        }
```

ESERCIZIO 6 - Soluzione

```
/* punto 2 */
    for (i=0; i<ns; i++)
        { if(SQUADRE[i].goal_fatti> SQUADRE[i].goal_subiti)
                printf("%s\n",SQUADRE[i].nome);
                printf("%d\n",SQUADRE[i].codice);
        }

/* punto 3 */
    printf("Inserisci un codice ");
    scanf("%d", &cod);
    i=0; T=0;
    while ((i < ns)&& (T==0))
      { if (SQUADRE[i].codice == cod)
                {printf("%s\n",SQUADRE[i].nome);
                 printf("%d\n",SQUADRE[i].goal_fatti);
                 printf("%d\n",SQUADRE[i].goal_subiti);
                 T=1;}
         i++;
     }
    if (T==0) printf("codice non trovato");
}
```

Esercizio 7

Si vuole implementare un programma per il calcolo dell'inflazione su determinati prodotti commerciali.

A tal scopo ogni prodotto è rappresentato tramite una struttura **`item`**, definita da una stringa **`name`** con il nome del prodotto, e da due float **`old_price`** e **`new_price`** rappresentanti i prezzi.

Esercizio 7

a) Si scriva una funzione **`lettura()`** che riceva come parametri di ingresso un vettore **`prezzi`** di strutture **`item`**, la dimensione fisica **`max`** del vettore **`prezzi`**, e un puntatore a intero **`num`** che rappresenta la dimensione logica del vettore. La funzione deve leggere da standard input il nome del prodotto ed i due prezzi, e deve copiare tale informazione nella prima posizione libera nel vettore **`prezzi`**.

Esercizio 7

La funzione deve terminare se l'utente inserisce come nome del prodotto il termine "fine", oppure se viene raggiunta la dimensione fisica del vettore.

La dimensione logica del vettore **`prezzi`** così riempito deve essere restituita tramite il parametro **`num`** (passato appunto per riferimento). Al termine della lettura dei dati la funzione deve restituire il valore 0.

Esercizio 7

b) Si scriva un programma **`main`** che, dopo aver definito un vettore di strutture **`item`** (di dimensione massima **`MAX_ITEM`**), invochi la funzione **`lettura()`** per riempire tale vettore.

Il programma stampi poi a video nome e tasso d'inflazione per ogni prodotto, utilizzando la formula:

$$infl_i = \left(\frac{new\_price_i}{old\_price_i} - 1 \right) * 100$$

Esercizio 7 - Soluzione

```
#include <stdio.h>
#include <string.h>

#define DIM 21
#define MAX_ITEM 100

typedef struct {
     char name[DIM];
     float old_price;
     float new_price;
} item;
...
```

Esercizio 7 - Soluzione

```
int lettura (item prezzi[], int max, int * num) {
    char name[DIM];
    *num = 0;

    printf("Inserire nome prodotto: ");   scanf("%s", name);

    while ((strcmp(name, "fine")) && (*num < max)) {
        strcpy(prezzi[*num].name, name);
        printf("Inserire old price: ");
        scanf("%f", &prezzi[*num].old_price);
        printf("Inserire new price: ");
        scanf("%f%*c", &prezzi[*num].new_price);

        (*num)++;

        printf("Inserire nome prodotto: ");
        scanf("%s", name);
    }  return 0;
}
```


Esercizio 7 - Soluzione

```
int main() {
    item V[MAX_ITEM];
    int num, i, result;
    float infl;

    result = lettura(V, MAX_ITEM, &num);

    if (result!=0) {
        printf("Problemi durante la lettura...\n");
        exit(-1);
    }

    for (i=0; i < num; i++) {
        infl = (V[i].new_price/V[i].old_price -1)*100;
        printf("Inflazione del prodotto %s: %6.2f%%\n", V[i].name, infl);
    }
    return 0;
}
...
```

Esercizio 8

Uno dei più antichi sistemi di codificazione di messaggi segreti si basa sulla sostituzione, secondo un certo ordine, dei caratteri componenti il messaggio.

Ad esempio, dato un messaggio composto dalle lettere:

{a, b, c}

E data una chiave di sostituzione che, per ogni lettera ne associa un'altra:

'a' → 'x'

'b' → 'y'

'c' → 'z'

Il messaggio originale può essere così riscritto:

{x, y, z}

Esercizio 8

Si vuole costruire un sistema di codifica/decodifica di questo tipo, facendo le seguenti assunzioni:

1. Le lettere componenti il messaggio sono tutte minuscole, ed i messaggi non possono contenere altri caratteri che lettere (no spazi, no numeri)
2. Il codice di sostituzione è dato da un array di 26 caratteri, che viene interpretato nel seguente modo: nella posizione ad indice 0 vi è il carattere che deve sostituire la lettera 'a', in posizione con indice 1 vi è il carattere che deve sostituire la lettera 'b', etc.

Esercizio 8

Si strutturi la soluzione implementando due funzioni:

```
void crypt( char source[],
            int length,
            char code[DIM_ALPHA],
            char dest[]);


void decrypt( char source[],
              int length,
              char code[DIM_ALPHA],
              char dest[]);
```

Ed infine si scriva un semplice main di prova.

Esercizio 8 - Soluzione

```
void crypt(char source[], int length, char
    code[DIM_ALPHA], char dest[]) {
       int i;

       for (i=0; i<length; i++) {
             dest[i] = code[source[i] - 'a'];
       }
}
```

Esercizio 8 - Soluzione

```
void decrypt(char source[], int length, char
    code[DIM_ALPHA], char dest[])
{
    int i;
    int j;
    int pos= -1;

    for (i=0; i<length; i++) {
       for (j=0; j<DIM_ALPHA && pos<0; j++) {
             if(source[i] == code[j])
                   pos = j;
       }
       dest[i] = 'a' + pos;
       pos = -1;
    }
}
```


Esercizio 8 - Soluzione

```
#define DIM 256
#define DIM_ALPHA 26

int main()
{
    char source[DIM] = "abc";
    char dest1[DIM] = {'\0',...};
    char dest2[DIM] = {'\0',...};
    char codice[DIM_ALPHA] = "cab";

    printf("ORIGINALE: %s\n", source);
    crypt(source, 3, codice, dest1);
    printf("CRIPTATO: %s\n", dest1);
    decrypt(dest1, 3, codice, dest2);
    printf("DE-CRIPTATO: %s\n", dest2);

    return 0; }
```