

// Stefano Volpe #0000969766# ①

// es.1: precondizione: "da è l'indice del primo elemento di vet su cui operare (incluso)  
// (in caso di azda: vettore vuoto) "a" è l'indice dell'ultimo elemento di vet su cui operare (incluso)

void es1 (int vet[], int da, int a)

{

if ( $a > da$ ) { // opera su almeno due elementi;

if ( $vet[da] \% 2 == 0$ ) // numero pari: in testa  
 $da = da + 1$ ;

else { // numero dispari: in coda

int t =  $vet[da]$ ;

$vet[da] = vet[a]$ ;

$vet[a] = t$ ;

$a = a - 1$ ;

}

es1(vet, da, a);

}

}

// postcondizione:

// la sottosequenza di vet che va dall'elemento di indice "da"  
// all'elemento di indice "a" inclusi soddisfa la richiesta  
// del testo

// Stefano Volpe #0006969766# ②

```
#include <cstring>
#define DIM 81
using namespace std;
```

```
struct Regalo {
```

```
    char nome[DIM];
    float prezzo; //in euro
    char destinatario[DIM];
    Regalo* next;
```

```
};
```

// es2a: precondizione: "lista" è una lista di regali (anche vuota)

// "n" e "d" hanno al più DIM caratteri (incluso '\0'), p > 0.0

```
Regalo* es2a(Regalo* lista, char n[], float p, char d[])
```

```
{    Regalo* res = new Regalo;
```

```
    strcpy(res->nome, n);
```

```
    res->prezzo = p;
```

```
    strcpy(res->destinatario, d);
```

```
    res->next = lista;
```

```
    return res;
```

// postcondizione: nella lista restituita è stato inserito (in testa) l'elemento

// segue nella prossima pagina...

// ... continua dalla pagina precedente: Stefano Volpetti#0000969766#③  
// es2b: preconditione: "lista" è una lista di regali (anche vuota),  
// "n" ha al più 10M caratteri (incluso '\0')  
float es2b(regalo\* lista, char n[])

{  
if (lista != NULL) {  
 if (strcmp(lista->nome, n) == 0) // nomi uguali  
 regalo\* coda = lista->next;  
 delete lista;  
 return es2b(coda, n);  
}  
}

// nomi diversi  
lista->next = es2b(lista->next, n);  
return lista;

{  
return NULL;  
} // postcondizione: restituisce il puntatore all'elemento alla lista da cui sono  
// eliminati TUTTI i regali con il nome specificato

// es2c: preconditione: "lista" è una lista di regali (anche vuota)  
// "d" ha al più 10M caratteri ('\0' incluso)  
float es2c(regalo\* lista, char d[])

{  
if (lista == NULL)  
 return 0.0;  
if (strcmp(lista->destinatario, d) == 0) // destinatari uguali  
 return lista->prezzo + es2c(lista->next, d);  
// destinatari diversi  
return es2c(lista->next, d);

} // postcondizione: restituisce il totale dei regali destinati a "d"

// Stefano Volpe # 0000969766 ES. 3

④

```
#include <cstring>
#define DIM 81
using namespace std;
```

```
class Automobile {
```

```
protected:
```

```
    char modello[DIM];
    float consumo_per_100km;
```

```
public:
```

```
    Automobile(char m[], float c) // prec: m'ha al più DIM caratteri (incluso '\0')
```

```
    {
```

```
        strcpy(modello, m);
```

```
        if (c > 0)
```

```
            consumo_per_100km = c;
```

```
        else
```

```
            consumo_per_100km = 1.0; // valore predefinito
```

```
}
```

```
    float calcola_consumi(float Km)
```

```
    { if (Km < 0) Km = 0.0; // valore predefinito;
```

```
        return consumo_per_100km / 100.0 * Km;
```

```
}
```

```
    float calcola_costo(float Km, float euro)
```

```
    { if (euro < 0) euro = 0.0; // valore predefinito
```

```
        return calcola_consumi(Km) * euro;
```

```
}
```

```
    //
```

```
// segue nella prossima pagina ...
```

// Stefano Volpe // 0000 969766//

// prosegue dalla scorsa pagina

class Benzina: public Automobile {  
protected:

float serbatoio; // in litri

public: Benzina(char m[], float c, float s) // in litri  
: Automobile(m, c) {

if (sc=0.0)

else serbatoio = 1.0; // valore predefinito

serbatoio = s;

float calcola-autonomia() // postcondizione: restituisce i Km

{

} return serbatoio / consumo-per-100km \* 100.0;

}

class Ibrida: public Benzina {

protected:

float risparmio;

public:

Ibrida(char m[], float c, float s, float t) // in litri e in percentuale

: Benzina(m, c, s) {

if (t<=0.0 || t>=100.0)

risparmio = 50.0; // valore predefinito

else

risparmio = t;

float calcola-consumi(float Km)

} return(100.0 - risparmio) / 100.0 \* Automobile::calcola-consumi(Km);

float calcola-costo(float Km, float euro)

} return (100.0 - risparmio) / 100.0 \* Automobile::calcola-costo(Km, euro);

float calcola-autonomia() // Postcondizione: restituisce i Km

} return 100.0 / (100.0 - risparmio) \* Automobile::calcola-autonomia();

// prosegue nell'ultimo foglio ...

6

// Stefano Volpe # 0000969766#  
class Elettrica: public Automobile {  
protected:  
    float batteria, l/kWh  
public:  
    Elettrica (char m[], float c, float b) l/kWh  
        : Automobile (m, c) {}  
        if (b <= 0.0)  
            batteria = 1.0; // valore predefinito  
        else  
            batteria = b;

} float calcola-autonomia () // postcondizione - restituisce i Km  
{  
    return batteria / consumo\_per\_100km \* 100;  
}

}