Esercizio 1 – Sintesi del codice (10.5 punti)

Si considerino le seguenti strutture dati: la struttura `prodotto` definisce un prodotto con il suo codice, il suo nome e il suo prezzo; `lista_prodotti` definisce una lista concatenata di prodotti; `lista_float` definisce una lista concatenata di numeri in virgola mobile.

```
#define MAX_NOME 15

typedef struct {
     unsigned int codice;
     char nome[MAX_NOME + 1];
     float prezzo;
} prodotto;

typedef struct nodo_prodotto_t {
     prodotto p;
     struct nodo_prodotto_t *next;
} nodo_prodotto;

typedef nodo_prodotto * lista_prodotti;

typedef struct nodo_float_t {
     float v;
     struct nodo_float_t *next;
} nodo_float;

typedef nodo_float * lista_float;
```

(a) Si scriva una funzione

```
lista_prodotti leggi_da_file(char *nome_file)
```

che acquisisce le descrizioni di alcuni prodotti da un *file* testuale il cui nome è passato come parametro e crea una lista dinamica di prodotti.
Ogni rigo del *file* contiene la descrizione di un prodotto come sequenza di:
codice prodotto (sequenza di caratteri indicante un numero intero positivo),
nome prodotto (sequenza di caratteri senza spaziature), e
prezzo in euro (sequenza di caratteri indicante un numero in virgola mobile con due cifre decimali);
un carattere di ***spazio*** è usato per separare ciascun valore. *(2.5 punti)*

Esempio del contenuto del *file* con la descrizione dei prodotti:

```
123456 Prodotto1 20.10
43216 Prodotto2 30.00
3212342 Prodotto3 40.85
32342 Prodotto4 10.00
1245 Prodotto5 100.50
```

Soluzione:

Usando una funzione ausiliaria per l'inserimento "in coda" di un nuovo elemento nella lista di prodotti:

```
lista_prodotti inserisciInCoda(lista_prodotti L, prodotto p) {

    nodo_prodotto *aux;

    if ( L == NULL ) {
        aux = (nodo_prodotto*) malloc(sizeof(nodo_prodotto));
        aux->p = p;
        aux->next = NULL;
        return aux;
    } else {
        L->next = inserisciInCoda(L->next, p);
        return L;
    }
}

lista_prodotti leggi_da_file(char *nome_file) {

    lista_prodotti L = NULL;
    FILE *fp = fopen(nome_file, "r");

    if (fp == NULL)
      fprintf(stderr, "Errore nell'apertura del file\n");
    else {
      prodotto p;
      while ( ! feof(fp) ) {
           fscanf(fp, "%u %s %f", &p.codice, p.nome, &p.prezzo);
           L = inserisciInCoda(L, p);
      }
      fclose(fp);
    }
    return L;
}
```

Soluzione alternativa

Usando un solo sottoprogramma con politica di "Inserimento In Testa" della lista di prodotti

```
lista_prodotti leggi_da_file(char *nome_file) {

    lista_prodotti L = NULL;
    FILE *fp = fopen(nome_file, "r");

    if ( fp == NULL )
      fprintf(stderr, "Errore nell'apertura del file\n");
    else {
      prodotto p;
      nodo_prodotto *aux;
      while ( ! feof(fp) ) {
        fscanf(fp, "%u %s %f", &p.codice, p.nome, &p.prezzo);
        aux = (nodo_prodotto*) malloc(sizeof(nodo_prodotto));
        aux->p = p;
        aux->next = L;
        L = aux;
      }
      fclose(fp);
    }
    return L;
}
```

(b) Si scriva una funzione iterativa

```
lista_prodotti elimina_prodotti(lista_prodotti L, float prezzo_minimo)
```

che modifica la lista passata come parametro rimuovendo tutti i prodotti che hanno un prezzo inferiore a `prezzo_minimo` e ritorna la lista modificata.
Durante la rimozione, prestare attenzione a liberare la memoria non più utilizzata. *(3 punti)*

Soluzione:

```
lista_prodotti elimina_prodotti(lista_prodotti L, float prezzo_minimo) {

    lista_prodotti ptr_precedente = NULL,
                   ptr_corrente = L,
                   testa = L,
                   temp;

    while (ptr_corrente != NULL) {

        if (ptr_corrente->p.prezzo < prezzo_minimo) {

          temp = ptr_corrente;
          if (ptr_precedente == NULL)
            testa = ptr_corrente->next;
          else
            ptr_precedente->next = ptr_corrente->next;
          ptr_corrente = ptr_corrente->next;
          free(temp);

        } else {

            ptr_precedente = ptr_corrente;
            ptr_corrente = ptr_corrente->next;

        }
    }

    return testa;
}
```

(c) Si scriva una funzione iterativa

```
lista_float somma(lista_prodotti L)
```

che genera una lista di numeri in virgola mobile a partire dalla lista di prodotti passata come parametro. L'elemento in posizione *i*-esima della lista di numeri in virgola mobile ha come valore la somma dei prezzi di tutti gli elementi della lista di prodotti passata come parametro fino alla posizione *i*. *(3 punti)*

Esempio:
Se la prima lista contenesse 4 prodotti con i seguenti prezzi:
`12.00  3.00  20.00  5.00`
La funzione costruirebbe una seconda lista contenete i seguenti valori:
`12.00  15.00(=12+3)  35.00(=12+3+20)  40.00(=12+3+20+5)`

Soluzione:

```
lista_float somma(lista_prodotti L) {

    lista_float testaListaFloat = NULL;
    nodo_float *ptr = NULL, *aux;
    float totaleCorrente = 0.0;

    while (L != NULL) {
        totaleCorrente += L->p.prezzo;
        aux = (nodo_float*) malloc(sizeof(nodo_float));
        aux->v = totaleCorrente;
        aux->next = NULL;
        if (ptr == NULL) {
          ptr = aux;
          testaListaFloat = aux;
        }
        else {
          ptr->next = aux;
          ptr = ptr->next;
        }
        L = L->next;
    }
    return testaListaFloat;
}
```