**TESTO**

**Esercizio 1**

Sia data la struttura seguente:

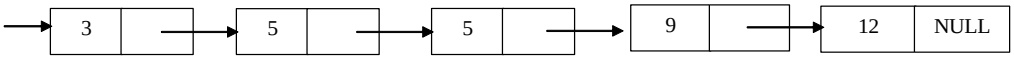
```
struct elem {int info; elem* pun;};
```

Scrivere una funzione che legge numeri interi da un file il cui nome è passato come argomento alla funzione e restituisce una lista di elementi di tipo `elem` che contiene un elemento per ogni numero intero positivo letto. La lista restituita può contenere duplicati e deve essere ordinata per valori crescenti del campo informazione.

Per esempio, se la funzione viene chiamata con il file seguente:

5 3 9 -8 5 12 eof

la funzione restituisce la lista:




**Esercizio 2**

Scrivere una funzione che prende in ingresso un vettore di interi di dimensione `n` e restituisce una matrice quadrata di interi di dimensione `nxn` inizializzata come segue. Tutti gli elementi di riga `i` e colonna `j>=i` sono uguali al valore dell'elemento del vettore nella posizione di indice `i`. Tutti gli elementi di colonna `j` e riga `i>=j` sono uguali al valore dell'elemento del vettore nella posizione di indice `j`. Le righe e le colonne sono numerate a partire da 0.

Per esempio, se la funzione viene chiamata con il vettore di dimensione `5` seguente:

```
3   1   7   9   2
```

la matrice restituita è (dimensione `5x5`):

```
3  3  3  3  3
3  1  1  1  1
3  1  7  7  7
3  1  7  9  9
3  1  7  9  2
```

**Esercizio 3**

Sia data la struttura seguente:

```
struct elem {char* nome; elem* pun;};
```

Scrivere una funzione ricorsiva che data una lista di stringhe composta da elementi di tipo `elem` ed una stringa `s`, restituisce il numero di occorrenze della stringa `s` nella lista.

**SOLUZIONE**

**Esercizio 1**

```
struct elem {int info; elem* pun;};
struct elem1 {char* nome; elem1* pun;};

elem* creaLista(const char* s){
elem* testa = NULL;
ifstream in(s);
if(!in)
  return testa;
int numero;
elem* p,*q, *r;

while (in >> numero) {
    if (numero >=0) {
    q = testa;
    for (q= testa; (q!= NULL) && (q->info <= numero);
                    q= q->pun)
        p=q;
   r= new elem;
   r->info = numero;
   r->pun = q;
   if (q==testa)
        testa = r;
   else
        p->pun = r;

  }
}
in.close();
return testa;
}

void stampa(elem* p) {
elem *q;
for (q=p; q!= NULL; q=q->pun)
    cout << q->info << ' ';
 }
```

**Esercizio 2**

```
 int* creamatrice(int v[], int n){
    int* m = new int [n * n];
    for (int i=0; i< n; i++){
      for (int j=i; j< n; j++)
        m[i*n +j] = v[i];
    }
    for (int j=0; j< n; j++){
      for (int i=j; i< n; i++)
        m[i*n +j] = v[j];
    }
    return m;
 }

void stampamatrice(int* mat, int dim){
for (int i=0; i< dim; i++){
    for (int j=0; j< dim; j++)
        cout << mat[i*dim +j] << ' ' ;
    cout << endl;
    }
  }
```

**Esercizio 3**

```
int conta (elem1* p0, const char* s){
 if (p0==NULL)
    return 0;
 if (strcmp(p0->nome,s)==0)
    return (1 + conta(p0->pun, s));
  else return conta(p0->pun, s);
}
```