Prova scritta 2 luglio 2015

Esercizio 1

```
void elimina (elem*& testa) {
 elem*p, *q;
 q= testa;
 while (q!=NULL) {
   if (q->info%2) {
        if (q==testa) {
              testa=test->pun;
              delete q;
              q=testa;
         }
       else {
            p->pun = q->pun;
            delete q;
           q=p->pun;
       }
    }
   else {
    p=q;
    q=q->pun;
   }
 } // fine while
}
```

Esercizio 2

v1 [3, 7, 25]
v2 [4, 5, 7, 42, 60]

v [60, 42, 25, 7, 7, 5, 4, 3]

Algoritmo 1: scorro v1 con indice i e v2 con indice j a partire dall'inizio e seleziono sempre il valore piu' piccolo. Inserisco tale valore in v a partire dal fondo usando l'indice k.
All'inizio: i=0, j=0, k= l1+l2 -1

```
v[k--] = (v1[i]<v2[j]) ? v1[i++]: v2[j++];
```

equivalente a

```
  if (v1[i]<v2[j]) {
        v[k] = v1[i];
        k--;
        i++;
     }
     else {                  // se v1[i] e' uguale a v2[j] copio l'elemento di v2
        v[k] = v2[j];
        k--;
        j++;
     }
```

```
int* creavettore (int* v1, int l1, int* v2, int l2)  {
 int* v = new int [l1+l2];
 int i, j, k;
 i=0;
 j=0;
 k= l1+l2 -1;

// caso generale ci sono ancora elementi da copiare in v1 e v2
while (i<l1 && j < l2)
        v[k--] = (v1[i]<v2[j]) ? v1[i++]: v2[j++];

// se ho finito di copiare v1, copio gli elementi rimasti di v2
if (i==l1)  {
        while (j<l2)
                v[k--]=v2[j++];
                }
```

```
// se ho finito di copiare v2, copio gli elementi rimasti di v1
else {
        while (i<l1)
                v[k--]=v1[i++];
 }
// scrivo il vettore nel file
// interi separati dal carattere spazio (‘ ‘)
ofstream  os(“output.txt”);

// controllo se ho errore nell’apertura del file
// in caso di errore stampo a video un messaggio di errore
// altrimenti scrivo nel file gli elementi del vettore
if (!os)
   cout << “errore nell’apertura del file \n”;
else
        for (i=0; i< l1+l2; i++)
                os << v[i] << ‘ ‘;

 // restituisco l’indirizzo del primo elemento del vettore (ilrtorno
// della funzione e’ di tipo int*
 return v;
}
```

Algoritmo 2: scorro v1 con indice i e v2 con indice j a partire dalla fine e seleziono sempre il valore piu’ grande. Inserisco tale valore in v a partire dall’inizio l’indice k.
All’inizio i=l1-1, j=l2-1, k= 0

```
int* creavettore (int* v1, int l1, int* v2, int l2)  {
 int* v = new int [l1+l2];
 int i, j, k;
 i=l1-1;
 j=l2-1;
 k= 0;

// caso generale ci sono ancora elementi da copiare in v1 e v2
while (i >= 0 && j >=0)
    if (v1[i]>v2[j]) {
          v[k] = v1[i];
          k++-;
          i--;
       }
```

```
      else {                  // se v1[i] e' uguale a v2[j] copio prima l'elemento di v2
          v[k] = v2[j];
          k++;
          j--;
       }

// se ho finito di copiare v1, copio gli elementi rimasti di v2
if (i==-1)  {
        while (j>=0)
                v[k++]=v2[j--];
   }
// se ho finito di copiare v2, copio gli elementi rimasti di v1
else {
        while (i>=0)
                v[k++]=v1[i--];
  }

ofstream  os("output.txt");
if (!os)
   cout << "errore nell'apertura del file \n";
else
        for (i=0; i< l1+l2; i++)
                os << v[i] << ' ';
 return v;
}
```

Esercizio 3

Algoritmo 1: funzione con argomenti la matrice, numero totale di elementi della matrice (9 se matrice 3x3)

Passo ricorsivo:
ultimo elemento + chiamata alla funzione con argomenti la matrice e numero di elementi -1
Caso base:
numero di elementi uguale a 0.

```
int conta (int* mat, int n) {
   if (n==0)
      return 0;
  return mat[n-1] + conta(mat, n-1);
}
```

Algoritmo 2: funzione con argomenti la matrice, dimensione della matrice e numero di elementi già sommati (k)

Passo ricorsivo:
elemento in posizione k + chiamata alla funzione con argomenti la matrice, la dimensione e numero di elementi già sommati +1
Caso base: numero di elementi sommati uguale al numero totale degli elementi della matrice
Chiamata alla funzione con argomento k uguale a 0.

```
int conta (int* mat, int dim, int k=0) {
   if (k==dim*dim)
      return 0;
  return mat[k] + conta(mat, dim, k+1);
}
```

Esercizio 4

Soluzione 4.1

0.75 è uguale a 0.5 + 0.25, dunque la sua rappresentazione su p=6 bit di cui f=5 per la parte frazionaria sarà 0.11000.
Riguardo al numero
-6.75: essendo negativo => s = 1
Passiamo a rappresentare la parte intera 6, vedendola come naturale 6: 110
La rappresentazione della parte frazionaria (0.75) è già stata calcolata.
A questo punto giustapponendo la rappresentazione della parte intera e quella della parte frazionaria otteniamo:
110.11
la cui rappresentazione normalizzata è
1.1011 per due elevato alla +2.
Dunque F sono i bit dopo la virgola, con l'aggiunta di 6 zeri:
F = 1011000000
Per trovare la rappresentazione di E basta trovare la rappresentazione del numero +2 in una rappresentazione con BIAS su 5 bit. Poichè il bias è 01111, basta sommare due: 10001.
E = 10001
In definitiva: s=1, F = 1011000000, E = 10001

Soluzione 4.2

Il programma calcola la distanza tra la rappresentazione ASCII
di 'a' (97 in decimale) e quella del carattere 'A' (65 in decimale)
che vale 32 (in base 10) e 20 in base 16. Pertanto la prima outbyte
stamperà a video "20". Il programma continua prelevando l'indirizzo di
memoria in cui inizia la stringa ASCII str, che viene memorizzato in EBX.
La prima istruzione output mostrerà a video il primo carattere
di str, ossia 'T'. Il programma prosegue caricando le codifiche ASCII dei
due caratteri successivi ('r' e 'e'), che vengono mostrate a video
dopo essere state incrementate di 0x20.
Pertanto il programma stamperà a video 'R' e 'E'.
Riassumendo il programma stampa a video la seguente stringa:
"20TRE"