Prova scritta 11 giugno 2015

Esercizio 1

```
void ins_ordinata (elem*& po, int n){
  elem *p, *q;  elem * r = new elem;
  r -> info =n;  r->pun = NULL;
  for (q=p0; q!=NULL && q->info < n; q=q->pun)
      p=q;
  if (q==p0)
     p0 = r;
  else
      p->pun = r;
r->pun = q;
 }

elem*  crealista( const char* nome) {
  ifstream in(nome);
  if (!in)  return NULL;
  elem* testa=0;
  int k;
  while (in >> k) {
       ins_ordinata(testa, k);
 }
char c;
in.clear();
if ((in >> c) && (c == '.'))
   return testa;
elem * q = testa:
while (testa!= NULL) {
   testa = testa->pun;
  delete q;
  q=testa;
 }
return NULL;
}
```

Esercizio 2

```
bool cerca(int* mat, int n, int m, int p){
 bool trovator=false;
 bool trovatoc=false;
 for(int i=0; i<n; i++)
   for(int j=0;  j<m-2; j++)
     if(mat[i*m+j] == p && mat[i*m+j+1] ==  p && mat[i*m+j+2]==p)
           trovator=true;
 if (!trovator)
   return false;
 for(int i=0; i<n-2; i++){
   for(int j=0;  j<m; j++)
     if(mat[i*m+j] == p && mat[(i+1)*m+j] ==  p && mat[(i+2)*m+j]==p)
           trovatoc = true;;
 return trovatoc;
 }
```

Esercizio 3

```
void stampa(int k) {
   if (k<=0)
         return;
    cout << "ciao ";
    stampa(k-1);
    cout << "mondo ";
}
```

Esercizio 4

4.1

```
La  rappresentazione in base 10 si trova utilizzando la "formula della sommatoria"
(ossia la definizione di notazione posizionale): 5*9^1+6*9^0 = 51.

Essendo 50 = 50+1 e 50 pari a 25*2, la rappresentazione in base 5 sarà  201.

Alla stessa conclusione si poteva arrivare applicando l'algoritmo standard mod/div:
q0 = 51
q1 = q0 div 5 = 10     a0 = q0 mod 5 = 1
q2 = q1 div 5 = 2      a1 = q1 mod 5 = 0
q3 = q2 div 5 = 0      a2 = q2 mod 5 = 2.
Dunque il numero e' a2a1a0: 201, cvd.
```

4.2

```
Siano X e Y la terzultima e l'ultima cifra del proprio numero
di matricola.
Il programma prima mette a 0 i 4 bit più significativi di %AL,
grazie all'AND con l'esadecimale 0x0F (0000-1111), lasciando
inalterati gli altri. Dunque a questo punto in AL c'è
la cifra meno significativa del proprio numero di matricola
(e pertanto la CALL output mostrerà a video "0Y").

Successivamente il programma stabilisce se Y è una potenza
di 2 (nel qual caso stamperà 'S') oppure no (nel qual caso
stamperà 'N').
Questo deriva dal fatto che un numero potenza di due,
una volta rappresentato in binario, avrà un solo bit ad uno
e tutti gli altri a zero.

La seconda parte del programma assembler infatti non
fa altro che contare il numero di bit ad uno.
Ogni volta che ne viene incontrato uno, viene incrementato
il contatore AH (che era stato precedentemente inizializzato a zero).
Se alla fine AH==1 il numero era una potenza di due (=> 'Y'),
altrimenti non lo era (=> 'N').

Riassumendo, nel caso di ingresso XY, l'uscita sarà:
input X0 uscita "00N" (infatti, 0 NON è una potenza di 2)
input X1 uscita "01S" (infatti, 1 è una potenza di 2:2^0)
input X2 uscita "02S" (infatti, 2 è una potenza di 2:2^1)
input X3 uscita "03N" (infatti, 3 NON è una potenza di 2)
input X4 uscita "04S" (infatti, 4 è una potenza di 2:2^2)
input X5 uscita "05N" (infatti, 5 NON è una potenza di 2)
input X6 uscita "06N" (infatti, 6 NON è una potenza di 2)
input X7 uscita "07N" (infatti, 7 NON è una potenza di 2)
input X8 uscita "08S" (infatti, 8 è una potenza di 2:2^3)
input X9 uscita "09N" (infatti, 9 NON è una potenza di 2)
```