Fondamenti di Informatica 1 – NETTUNO – Andrea Sterbini
Svolgimento del compito d'esame del 14-1-06

Esercizio 1 (conversione da numeri romani a arabi)

Ogni lettera del numero romano ha il suo valore, tranne se è seguita da lettere di valore maggiore. Quindi, con l'aiuto della funzione di appoggio cifraRomana otteniamo.

```
int cifraRomana ( char lettera ) {
    switch (lettera) {
    case 'M': return 1000;
    case 'D': return 500;
    case 'C': return 100;
    case 'L': return 50;
    case 'X': return 10;
    case 'V': return 5;
    case 'I': return 1;
    default: return 0;
    }
}
int romano2arabo( char * romano ) {
    int i, arabo = 0;
    for (i=0 ; romano[i] ; i++) {
        if ( cifraRomana(romano[i]) < cifraRomana(romano[i+1]))
            arabo -= cifraRomana(romano[i]);
        else
            arabo += cifraRomana(romano[i]);
    }
    return arabo;
}
```

Esercizio 2 (triangolo di Tartaglia)

```
#define NMAX 20
void triangoloTartaglia( int matrice[NMAX][NMAX], int N) {
    int i, j;
    for (i=0 ; i< NMAX  ; i++)
        for (j=0 ; j< NMAX ; j++)
            matrice[i][j] = 0;          // azzero la matrice
    for (i=0 ; i< NMAX  ; i++) {
        matrice[i][0] = 1;              // inserisco il primo
        for (j=1 ; j< NMAX ; j++)       // calcolo gli altri
            matrice[i][j] = matrice[i-1][j] + matrice[i-1][j-1];
    }
}
int main() {
    int M[MAX][MAX];
    int N;
    printf("Numero di righe da calcolare (tra 1 e %d): ", MAX);
    scanf("%d", &N);
    triangoloTartaglia(M, N);
}
```

Esercizio 3 (funzione ricorsiva)

L'unica cosa da notare è che la sottostringa centrale si ottiene spostandoci di un carattere in avanti. Si assume che la variabile lunghezza indichi correttamente il numero di caratteri della stringa per cui non è necessario esaminare o inserire lo zero terminatore di stringa.

Per controllare se i due caratteri sono uno maiuscolo e l'altro minuscolo si può usare la funzione isupper ma anche il trucco usato da alcuni di voi nel loro elaborato (vedi sotto).

```
#define TRUE 1
#define FALSE 0
int palindromaInversa( char * stringa, int lunghezza) {
    if (lunghezza <= 2)
        return TRUE;
    else{
        if (abs(stringa[0]-stringa[lunghezza-1]) != abs('A' - 'a'))
            return FALSE;
        else
            return palindromaInversa( stringa + 1, lunghezza - 2);
    }
}
```

Esercizio 4 (codifica dei numeri)

I due numeri M ed N in decimale sono

M = $AF08_{16}$ = 10*16^3+15*16^2+0*16^1+8*16^0 = **44808_{10}**

N = $5E5_{16}$ = 5*16^2 + 14*16^1 + 5*16^0 = **1509_{10}**

La loro rappresentazione binaria naturale è

M = 1010 1111 0000 1000 servono 16 bit

N = 0101 1110 0101 servono 11 bit

La rappresentazione in complemento a 2 su 24 bit si ottiene estendendo il segno

M = 0000 0000 1010 1111 0000 1000

N = 0000 0000 0000 0101 1110 0101

Per ottenere -N da N bisogna **calcolare l'opposto** del numero, ovvero:

- complementare il numero 1111 1111 1111 1010 0001 1010
- sommare 1 1111 1111 1111 1010 0001 1011 = -N

La somma di 3M-N si ottiene:

M 0000 0000 1010 1111 0000 1000 +

2M <u>0000 0001 0101 1110 0001 0000</u> =

3M 0000 0010 0000 1101 0001 1000 +

-N <u>1111 1111 1111 1010 0001 1011</u> =

0000 0010 0000 0111 0011 0011 = 020733_{16} = 132915_{10}