Università di Roma 1 – Corso di laurea in Ingegneria – Consorzio Nettuno
Esame di Fondamenti di Informatica 1 – 18-3-06 – Andrea Sterbini

Esercizio 1 (stringhe)

Si scriva la funzione C col prototipo: **`int leggi_in_base(char* N, int X)`** che riceve una stringa **`N`** che rappresenta un numero codificato nella base **`X`** (compresa tra 2 e 35) e torna il valore del numero **`N`**

NOTA: le cifre di valore maggiore a 10 sono codificate dalle lettere da A a Z come segue

```
A = 10, B = 11, C = 12, D = 13, ..., X = 23, Y = 24, Z = 35
```

ESEMPIO: **`leggi_in_base( "0A14", 12)`** torna il valore 1456

Esercizio 2 (stampa)

Si scriva la funzione C col prototipo: **`void stampa_in_base(int N, int X)`** che:

- stampa il valore del numero **`N`** convertito in base **`X`** (compresa tra 2 e 35)

ESEMPIO: **`stampa_in_base( 1456, 3)`** stampa la stringa **`1222221`**

Esercizio 3 (matrice)

Si scriva la funzione C che si chiama **`somma_a_farfalla`** che riceve come argomenti:

- una matrice quadrata interi di dimensione N x N
- il valore intero N che indica la dimensione della matrice

e calcola la somma degli elementi che si trovano sulle due diagonali e lungo i bordi verticali come in figura

ESEMPIO: la somma della matrice in figura vale **370**

1	2	3	4	5	**6**
7	**8**	9	10	**11**	**12**
13	14	**15**	**16**	17	**18**
19	20	**21**	**22**	23	**24**
25	**26**	27	28	**29**	**30**
31	32	33	34	35	**36**

Esercizio 4 (funzione ricorsiva)

Si sviluppi la funzione ricorsiva **`F`** definita come segue:

- F(N) = 0 se N = 1
- F(N) = F(N/2)+1 se N è pari
- F(N) = F(3*N+1)+1 se N è dispari

Si svolga l'andamento delle chiamate ricorsive e del ritorno dei risultati per N=10

Esercizio 5 (rappresentazione dei numeri)

Siano dati i due numeri ottali **`M = 2362`** ed **`N = 412`**

- Si calcoli in questa codifica il valore **`Y = 5 * N - M`**
- Si indichi per ciascuno dei tre numeri **`M,N`** e **`Y`** di quanti bit al minimo si ha bisogno per rappresentarlo nella codifica binaria col segno e lo si rappresenti in tale codifica
- Si indichi di quali numeri **decimali** si tratta
- Si mostri qual'è la codifica binaria in complemento a 2 con **16 bit** dei due numeri **`-N`** ed **`M`**

Svolgimento

Esercizio 1

Per calcolare un numero in una data base basta moltiplicare il valore di ciascuna cifra per la base elevata alla potenza che corrisponde alla posizione della cifra nella stringa contando da destra (partendo da 0 per il carattere che si trova all'estremo destro). Quindi se la base è **b**, la stringa "$a_n a_{n-1}...a_2 a_1 a_0$" vale

$$a_n a_{n-1}...a_2 a_1 a_0 = (b^n a_n + ... + b^1 a_1 + b^0 a_0) = ((...(((a_n * b)+a_{n-1})*b) ... + a_1)*b + a_0)$$

Quindi una possibile soluzione è (usando la formula sulla destra):

```
// funzione di utilità che calcola il valore di un carattere
int valore(char cifra, int base) {
  if (cifra <= '9')  return cifra – '0';            // valore delle cifre decimali
  else               return (cifra – 'A') + 10;     // valore delle lettere
}
int leggi_in_base(char* N, int base) {
  int risultato = 0, i;
  for (i=0 ; N[i] != '\0' ; i++) {
    risultato *= base;
    risultato += valore(N[i], base);
  }
  return risultato;
}
```

Esercizio 2

Per stampare un numero in una certa base dobbiamo calcolare i resti delle divisioni del numero per la base e stamparli in ordine opposto. Per evitare di usare un vettore per memorizzare le cifre prima della stampa, possiamo scrivere la funzione come funzione ricorsiva che:

- esce senza stampare se il numero è zero
- altrimenti divide il numero per la base e ne calcola il resto (cioè la cifra meno significativa)
- poi chiama ricorsivamente se stessa sul risultato della divisione per stampare le cifre più significative
- infine stampa il carattere meno significativo che corrisponde al resto precedentemente calcolato

Quindi una possibile implementazione è:

```
char * cifre = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
void stampa_in_base(int numero, int base) {
  if (numero == 0)
    return;
  int resto = numero % base;          // calcolo la cifra meno significativa
  stampa_in_base(numero/base, base);  // stampo le altre cifre
  printf("%c", cifre[resto]);         // stampo la cifra meno significativa
}
```

Esercizio 3

Per evitare di sommare più volte gli stessi elementi scandisco tutti gli elementi della matrice e sommo solo quelli che "sono su una delle diagonali o sui bordi verticali".
Si noti che la matrice ha dimensione non fissata, per cui va passata come un puntatore, e che quindi gli elementi vanno estratti dalla memoria calcolandone la posizione e dereferenziando il puntatore ottenuto.

```
somma_a_farfalla(int * matrice, int N) {
  int i, j, somma=0;
  for (i=0 ; i<N ; i++)                          // ciclo sulle righe
    for (j=0 ; j<N ; j++)                        // ciclo sulle colonne
      if (i==0 || i==N-1 || i==j || i+j==N-1)    // test "farfalla"
        somma += *(matrice + i*N + j);           // trovo l'elemento
  return somma;
}
```

Esercizio 4

La funzione è presto implementata:

```
int F(int N) {
   if (N == 0)      return 0;
   if (N%2 == 0)    return 1+F(N/2);
   else             return 1+F(3*N+1);
}
```

Lo svolgimento delle chiamate e dei risultati di F(10) è:

F(10)
F(5)
F(16)
F(8)
F(4)
F(2)
F(1) = 0
1+0=1
1+1=2
2+1=3
3+1=4
4+1=5
5+1=6

Quindi F(10)=6

NOTA: la funzione conta il numero di passaggi necessari per arrivare al valore 1 se si divide un numero per due quando è pari e lo si moltiplica per 3 e si somma 1 quando è dispari. Esiste una congettura (che è stata controllata anche per valori di N molto grandi) secondo la quale per ogni N la funzione termina sempre (ovvero si raggiunge sempre il valore di N=1), ma non è stata ancora dimostrata.

Esercizio 5

Per calcolare Y bisogna calcolare (in base 8):

$\mathbf{Y = 5 * N - M = 5_8 * 412_8 - 2362_8 = 2462_8 - 2362_8 = 100_8}$

La codifica binaria col segno di lunghezza minima si ottiene da un numero ottale rappresentando ciascuna cifra come un numero binario di 3 cifre (cioè i valori da 0 a 7), togliendo gli zeri a sinistra e precedendo il tutto con un bit di segno (che in questo caso è sempre 0 Perché tutti e 3 sono positivi):

Y = 0 1 000 000 N = 0 100 001 010 M = 0 10 011 110 010

Quindi il numero minimo di bit necessario è: Y -> 8 N -> 10 M -> 12

I valori decimali dei tre numeri sono:

$Y = 64_{10}$

$N = 4*64+1*8+2 = 256+8+2 = 266_{10}$

$M = ((2*8+3)*8+6)*8+2 = (19*8+6)*8+2 = 158*8+2 = 1264+2 = 1266_{10}$

La codifica in complemento a 2 su 12 bit dei numeri M e -N si ottiene:

M è positivo quindi la sua codifica in complemento a 2 su 12 bit è la stessa (con i bit raggruppati 4 a 4)

M -> 0100 1111 0010
per ottenere la codifica in complemento a 2 su 16 bit dobbiamo estenderne il segno, copiando a sinistra il bit più significativo fino a riempire i 16 bit
M -> 0000 0100 1111 0010

applicando lo stesso procedimento otteniamo N -> 01 0000 1010 -> 0000 0001 0000 1010

per ottenere -N calcoliamo l'inverso di N, quindi:

complementiamo N e gli aggiungiamo 1

-N = 1111 1110 1111 0101 + 1 = 1111 1110 1111 0110