Esempi di invarianti 2

albero di Natale: si legge n >2 e si stampa l'albero di altezza n

n=3

```
  *
 ***
*
```

n=4

```
   *
  ***
 *****
*
```

n=5

```
    *
   ***
  *****
 *******
*
```

ogni riga consiste di alcuni spazi seguiti da stelle:
nella prima riga n-2 spazi seguiti da 1 stella
nella seconda riga n-2-1 spazi seguiti da 1+2 stelle
nella terza riga n-2-1-1 spazi seguiti da 1+2+2 stelle
e questo per n-1 righe
poi c'è la riga del tronco con 1 sola stella

useremo 2 variabili: nspazi e nstelle è facile definisre cicli che stampano nspazi e nstelle su una riga:

```
int i=0;
while(i <nspazi)  //R=(0<=i<=nspazi) && (stampati i spazi)
{
   cout<<‘ ‘;
   i++;
}
```

POST=(stampati nspazi spazi)

in uscita dal ciclo (i = nspazi)

un ciclo simile stampa le stelle

soluzione : serve un ciclo sulle n-1 righe dell'albero che faccia variare nspazi e nstelle in modo appropriato per ciascuna riga e che usi i cicli appena visti per stampare spazi e stelle della riga.

```
int n;  cin >> n;
int k=0, nspazi=n-2, nstelle=1, i=0;
while (*) (k < n-1)
{
  i=0;
  while (i<nspazi) {cout<<‘ ; i++;}
  i=0;
  while(i<nstelle) {cout <<‘*’; i++;}
  cout<<‘\n’;
  nspazi--;     nstelle=nstelle+2;     k++;
} POST= stampate le prime n-1 righe dell’albero di altezza n
```

R=(0<=k<=n-1)&&(stampate le prime k righe) &&
(nspazi e nstelle sono ok per prossima riga)

R=(0<=k<=n-1)&&(stampate le prime k righe) &&
(nspazi e nstelle sono ok per prossima riga)

1) con k=0, nspazi =n-2 e nstelle=1 vale R perché ok per la prima riga
2) il corpo stampa la prossima riga e poi prepara nspazi e nstelle per quella successiva

3) R && (k =n-1)

=> POST= stampate le prime n-1 righe dell'albero di altezza n

Nuovo esempio: leggere interi da cin fino alla sentinella -1 e sommarli in somma:

//PRE=(cin contiene -1 eventualmente preceduto da altri interi)

R=(letti m interi, con m >=0) &&
(continua => sono tutti diversi da -1 e somma è la loro somma) && (se !continua => l'ultimo letto è -1 e somma è la somma dei precedenti che sono tutti diversi da -1)

```
int x, somma=0;
bool continua=true;
while (*) (continua)
{ cin >> x;
   if (x != -1)
      somma=somma+x;
    else
      continua=false;
}
```

//POST=(somma contiene la somma degli interi che precedono il primo -1 in cin)

Dimostrazione:

1) con m=0 e somma=0, vale R

2) invarianza:

R && continua => mai letto -1 e somma è somma dei letti. Faccio una lettura e ci sono 2 casi:

-letto -1 =>continua = false

-non letto -1 => lo sommo a somma

$\Rightarrow$ vale R di nuovo

R=(letti m interi, con m >=0) &&
(continua => sono tutti diversi da -1 e somma è la loro somma) &&
(se !continua => l'ultimo letto è -1 e somma è la somma dei precedenti che sono tutti diversi da -1)

3) R && !continua => POST= somma contiene la somma degli interi che precedono il primo -1 in cin)

altra soluzione per lo stesso esercizio:

//PRE=(cin contiene -1 eventualmente preceduto da altri interi)

```
int somma=0, x;
cin >> x;
while(x != -1)
{
  somma=somma+x;
  cin >> x;
}
```

R=(letti m+1 valori) &&(i primi m sono diversi da -1 e somma è la loro somma) &&(l'ultimo è in x)

//POST=(somma contiene la somma degli interi che precedono -1 in cin)

R && (x = -1) => POST

estendiamo l’esercizio: la sentinella resta -1, ma, se -1 è nei primi10 numeri, sommiamo quelli che precedono -1, altrimenti sommiamo i primi 10 numeri

a voi di risolverlo e possibilmente in 2 modi, seguendo i 2 esempi di programmi appena esaminati