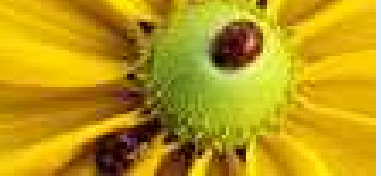
Olimpiadi dell’informatica

Rosario Culmone

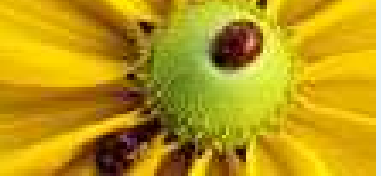
rosario.culmone@unicam.it



Cosa sono le olimpiadi dell'informatica

Le Olimpiadi internazionali dell'informatica, in inglese International Olympiad in Informatics e in acronimo IOI, sono competizioni fra studenti delle scuole secondarie che si svolgono annualmente. Le prime di queste competizioni si sono svolte nel 1989 e sono state indette dall'UNESCO. La prossima sarà tenuta in Egitto. In Italia le Olimpiadi Italiane dell'Informatica (OII) sono gestiste dall'AICA (http://www.aica.it)

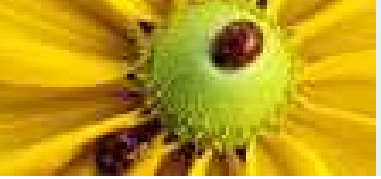
Lo scopo delle olimpiadi è diffondere mediante la competizione la cultura informatica tra tutti i paesi del mondo.



Alcuni link

Il comitato internazionale è su http://www.ioinformatics.org/
L'organizzazione è su http://olympiads.win.tue.nl/ioi/
Il comitato italiano è su http://www.olimpiadi-informatica.it/
Coordinatore nazionale Roberto Grossi http://www.di.unipi.it/ grossi/
Allenamenti http://allenamenti.olimpiadi-informatica.it/
Il sito della classe di informatica a Camerino: http://www.cs.unicam.it

La mia posta elettronica: rosario.culmone@unicam.it



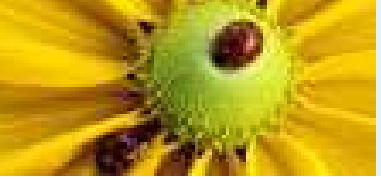
Prova internazionale: come si svolge

La competizione internazionale si svolge nel seguente modo:

- Due giorni di competizione
- Tre problemi in 5 ore
- Ogni studente lavora da solo con un computer (no libri)
- Si programma in C/C++ o Pascal

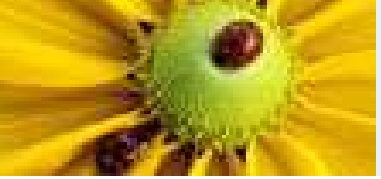
La valutazione avviene:

- 10 o 20 prove su dati segreti
- punteggio per ogni prova e rispetto dei limiti (tempo e/o spazio)
- uso di librerie per accedere ai dati
- esiste la possibilità di prove che non richiedono la presentazione del programma ma dell'output del programma
- in questo ultimo caso teoricamente si può provare a rispondere senza scrivere un programma



Vincitore

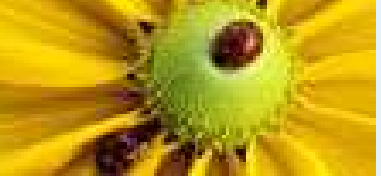
- la somma dei punti per ogni prova dei due giorni determina una graduatoria generale
- viene premiata la metà migliore (due per ogni paese)
- il rapporto delle medaglie è (oro:argento:bronzo:nessuna) 1:2:3:6



Prerequisiti 1

Conoscenza di un linguaggio Avere una eccellente conoscenza di un linguaggio di programmazione tra C o Pascal. Il linguaggio è utilizzato come strumento per poter codificare algoritmi complessi che sono la vera sfida. Il programma più lungo raramente supera le 50 righe!

Tecniche fondamentali di programmazione È necessario conoscere l'input/output da file nei dettagli, essere in grado di gestire memoria dinamicamente (allocazione e deallocazione), e sapere scrivere funzioni ricorsive (funzioni che chiamano se stesse). La ricorsione è fondamentale per la soluzione di problemi enumerativi o di ricerca esaustiva di soluzioni ottime.

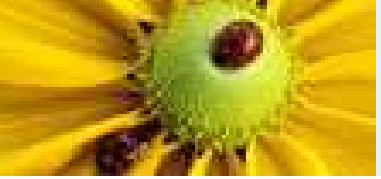


Prerequisiti 2

Strutture dati elementari Liste (a singolo e doppio concatenamento), alberi binari e i relativi algoritmi di base devono essere noti.

Algoritmi fondamentali È necessario conoscere bene almeno un algoritmo di ordinamento (meglio se di ordine $nlog(n)$ e la ricerca dicotomica in strutture ordinate.

Materiale didattico C'è molto materiale didattico in linea sull'argomento; ad esempio, le completissime dispense di Giovanni Pighizzini `http://homes.dico.unimi.it/~pighizzi/prog1999/lezioni` per il linguaggio Pascal e le dispense di Paolo Liberatore `http://www.dis.uniroma1.it/~liberato/struct/dispensa.shtml` per il linguaggio C. In ogni caso qualsiasi manuale di C o Pascal va bene.



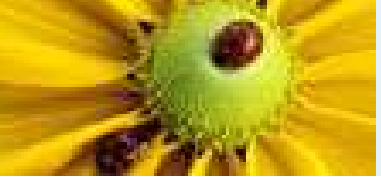
Strumenti

Le conoscenze fondamentali sono quelle algoritmiche. Dando per scontato che un partecipante sa utilizzare perfettamente tutte le strutture di basi fondamentali (vettori, liste ecc.), è necessaria anche una conoscenza reale di alberi, tavole di hashing, grafi orientati e non e dei relativi algoritmi di base.

Pe quanto riguarda gli strumenti (Sistema operativo)

- Fino al 2000 si gareggiava con DOS
- Nel 2001 sono stati resi disponibili Linux e Windows
- Dal 2002 l'ambiente ufficiale di gara è Linux

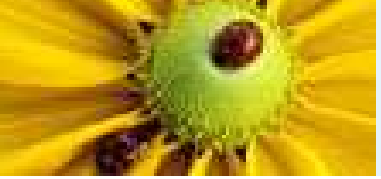
Quindi bisogna conoscere un editore e un compilatore C o Pascal per Linux (GNU C, Pascal). Sono disponibili dispense su http://allenamenti.olimpiadi-informatica.it/ su cosa è necessario sapere di Linux per competere.



Standard

Sia il linguaggio C che il Pascal sono nella versione ISO (ANSI C). Non possono essere utilizzate librerie non standard o dipendenti dal sistema operativo. Nel caso del C le librerie (ANSI C) ammesse sono:

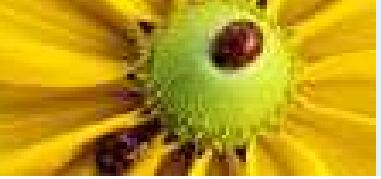
```
#include <stdio.h>      /* letture e scritture di flussi */
#include <ctype.h>      /* gestione dei caratteri */
#include <stdlib.h>     /* dichiarazioni varie */
#include <locale.h>     /* dipendenze locali (-)*/
#include <float.h>      /* costanti in virgola mobile locali (-) */
#include <stdarg.h>     /* parametri sulla riga di comando (-)*/
#include <stddef.h>     /* costanti di uso comune */
#include <limits.h>     /* caratteristiche fisiche della machina (-)*/
#include <assert.h>     /* macro per debug (-)*/
#include <strings.h>    /* operazioni su stringhe */
#include <math.h>       /* libreria matematica */
#include <errono.h>     /* codici di errore (-)*/
#include <setjamp.h>    /* salti non locali (-)*/
#include <signal.h>     /* processi (-)*/
#include <time.h>       /*data e ora (-)*/
```



Allenamenti on line

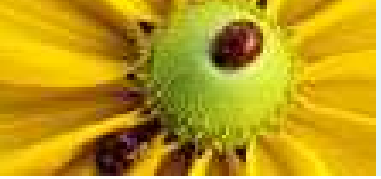
Per quanto riguarda gli allenamenti è importante mettersi alla prova cercando di risolvere esercizi sempre più difficili scaricandoli da http://allenamenti.olimpiadi-informatica.it/.
Sempre su http://allenamenti.olimpiadi-informatica.it/ è disponibile un test onLine per iniziare.

Gli algoritmi e la codifica è orientata alla rapida soluzione, normalmente non è richiesta efficienza, eleganza, compattezza della codifica.



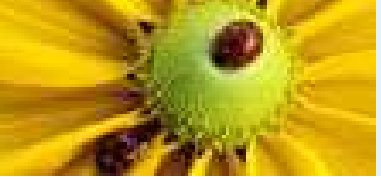
Esempio di problema: Palindrome

Una stringa è palindroma se è simmetrica, cioè se è identica quando viene letta da sinistra a destra o da destra a sinistra. Dovete scrivere un programma che, data una stringa, determini il minimo numero di caratteri da inserire nella stringa allo scopo di renderla palindroma. Ad esempio, inserendo due caratteri nella stringa Ab3bd la si può transformare in una palindroma (dAb3bAd o Adb3bdA). Ciononostante, non è possibile ottenere una stringa palindroma inserendo meno di due caratteri.



Palindrome: dati di input

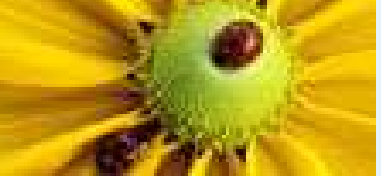
Il file di input si chiama PAL.IN. La prima riga del file contiene un numero intero N , $3 \leq N \leq 5000$, che è la lunghezza della stringa. La seconda riga contiene una stringa di lunghezza N. La stringa è formata da lettere maiuscole $('A' \ldots' Z')$, lettere minuscole $('a' \ldots' z')$ e cifre $('0' \ldots' 9')$. Le lettere maiuscole e minuscole devono essere considerate diverse.



Palindrome: dati di output

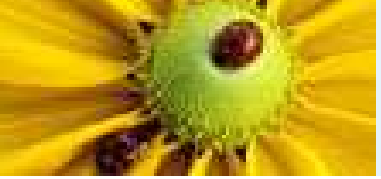
file di output si chiama PAL.OUT. La prima riga contiene un numero intero, che è il minimo numero di caratteri richiesto. Esempio di input e output

PAL.IN	PAL.OUT
5 Ab3bd	2



Palindrome: soluzione

```
#define min(a,b) ((a)<(b)?(a):(b))
#define MAXLUN 5000
#define MINLUN 3
char parola[MAXLUN];  int ott[3][MAXLUN];  FILE *F;

int main(int argc, char *argv[]) {
 int k, i, N; FILE *F; F = stdin;
 fscanf(F, "%d\n", &N);
 for (i=0; i<N; i++) fscanf(F, "%c", &parola[i]);
  for(k=2; k<=N; k++)
   for(i=0; i<=N-k; i++) {
    if (parola[i]==parola[i+k-1]) ott[(k+2)%3][i]=ott[k%3][i+1];
    else ott[(k+2)%3][i]=1+min(ott[(k+1)%3][i],ott[(k+1)%3][i+1]);
  }
 F = stdout; fprintf(F, "%d\n", ott[(N+2)%3][0]);
 return 0;
}
```



Pregara

Due aspetti importanti:

Fondamentali , ovvero saper esprimere senza errori e rapidamente gli algoritmi in un linguaggio di programmazione (C o Pascal)

Vedere gli algoritmi , ovvero allenare la mente a trovare soluzioni algoritmi a problemi complessi

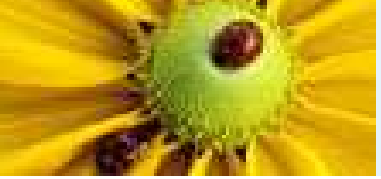
Ambedue gli aspetti si potenziano con l'allenamento che consiste in:

Fondamentali

- Leggere esempi di programmi
- Imparare lessico, sintassi e semantica del linguaggio di programmazione
- Conoscere le principali strutture dati e gli algoritmi fondamentali

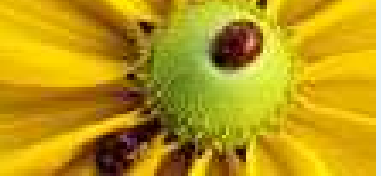
Vedere gli algoritmi

- Leggere esercizi svolti e comprenderne i "trucchetti"
- Classificare gli esercizi per tipologia "algoritmica"
- Provare a svolgerli per complessità magari individuando solo l'algoritmo e poi codificandolo.



Competizione

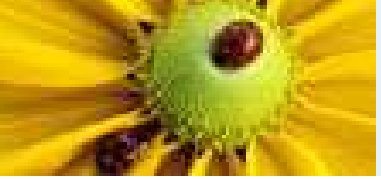
- Si tratta di una vera e propria competizione ma sopra tutto una sfida con se stessi
- I risultati possono venire con il tempo
- Da molta soddisfazione "vedere" cose che gli altri "non vedono"
- Al contrario di come si crede problemi complessi talvolta richiedono soluzioni semplici ma "astute". Difficilmente una prova richiede più di 50 righe di programma.



Babilonesi

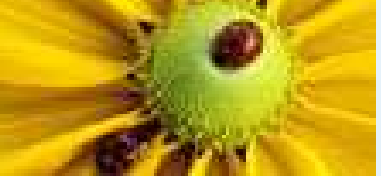
I babilonesi (3500 a.C.) hanno lasciato molte tavolette di argilla con algoritmi per il calcolo di superfici, quantità, dimensioni, ecc. Erone, vissuto nel I sec. a.C descrive un metodo per l'estrazione della radice quadrata che alcuni fanno risalire ai babilonesi

Un esempio è l'algoritmo per la costruzione di un quadrato e quindi per il calcolo di $\sqrt{2}$. Si basa di un algoritmo di approssimazione estremamente potente (3 iterazioni producono 1,41421, ovvero precisione alla quinta cifra decimale). Come molti metodi antichi si basa sull'utilizzo di strumenti semplicissimi: pali e corde. Si pensa che il metodo servisse per costruire edifici quadrati o per tracciare appezzamenti di terra.



Numeri primi

Un altro problema ancor oggi di rilevante importanza è il calcolo dei numeri primi. Per sapere se un numero intero n è primo o no bisogna provare che non è divisibile per nessun numero $k < n$. Naturalmente k non è 1. Bisogna effettuare n divisioni e controllare che tutti hanno resto! Eratostene di Cirene (284 a.C. - 192 a.C.) propose un algoritmo per trovare tutti i numeri primi da 1 a n.



Crivello di Eratostene

Se si dispongono n interi su una tabella:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...	n

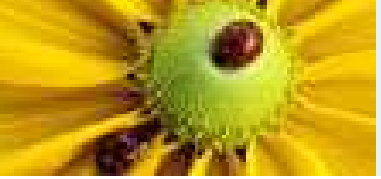
Partendo da 2 e a distanza 2 si cancellano tutte le cifre maggiori di 2 sino a n

1	2	3		5		7		9		11		13		15		17	...	n

Si considera il primo numero incontrato dopo 2 nella nuova lista, è 3 e si cancellano tutti i numeri a distanza 3

1	2	3		5		7				11		13				17	...	n

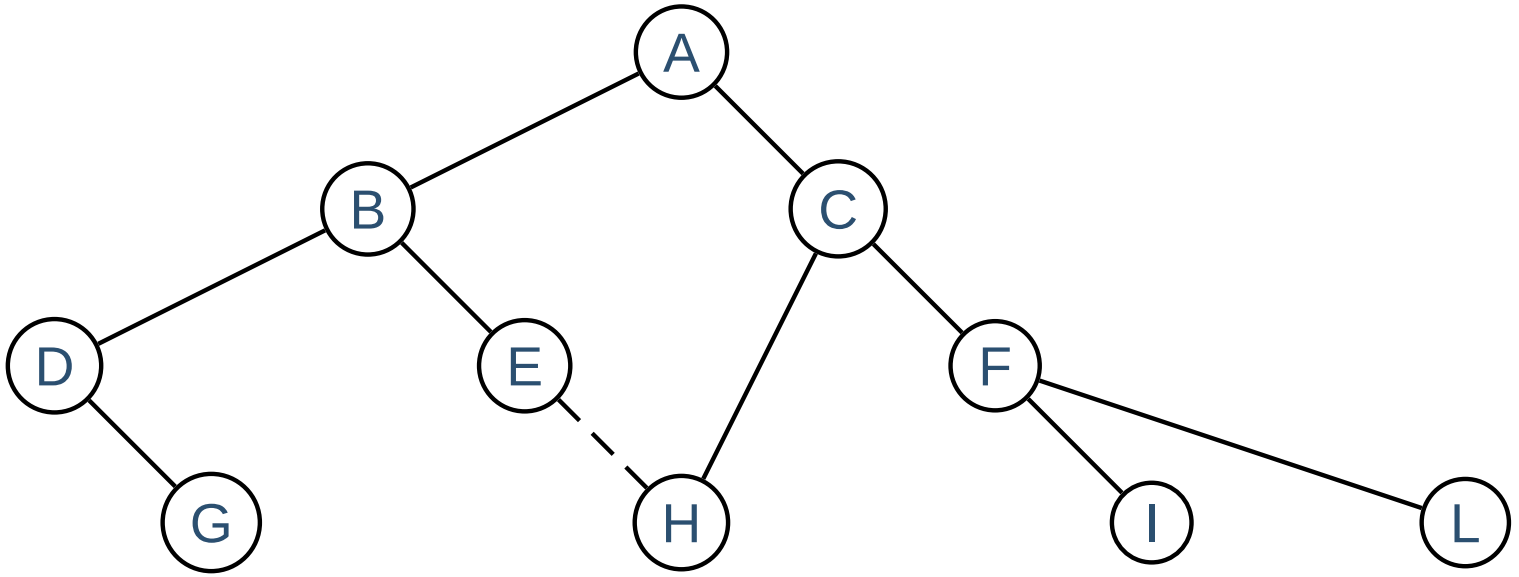
Si ripete l'operazione precedente sino a quando non si raggiunge n. I numeri rimasti sulla lista sono solo numeri primi.

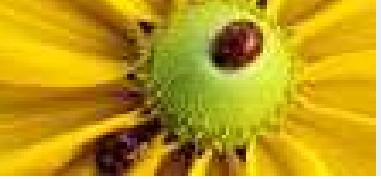


Visite di grafi

La conoscenza di tecniche standard permette di trovare rapidamente una soluzione.

Ad esempio dato un albero genealogico, determinare se esistono anomalie ovvero se esiste un nodo con due padri.

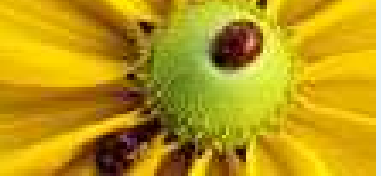




Soluzione in C

Supponiamo per semplicità che ogni persona possa avere al più due figli. Ogni persona può essere identificata con una etichetta e un valore intero inizializzato per tutti a FALSE. Si ha che:

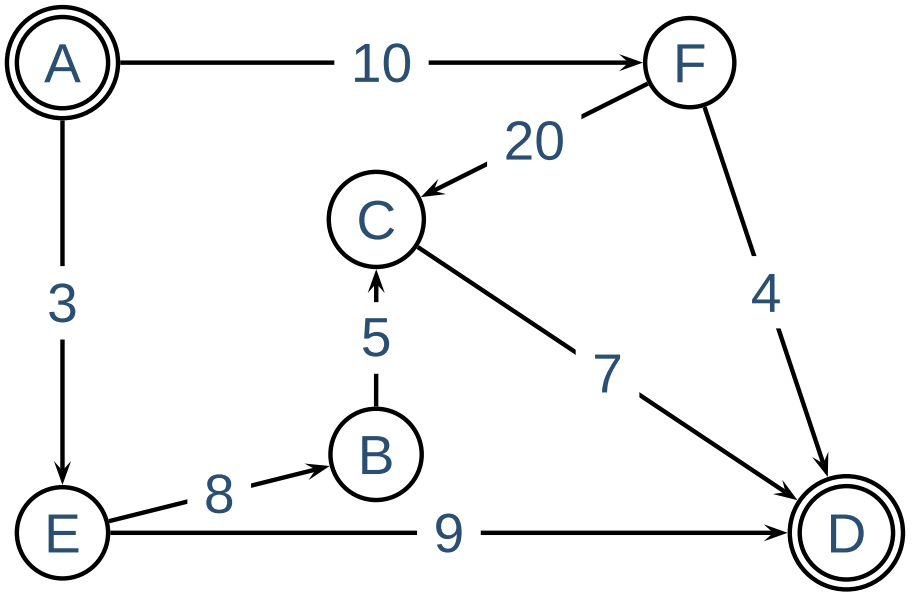
```
#define TRUE 1
#define FALSE 0
typedef struct node {
  char n; int v; struct node *x, struct node *y }; NODE;

int find(NODE *r) {
  if (r == NULL) return FALSE;
  else if (r->v == TRUE) return TRUE;
    else {r->v=TRUE; return find(r->x) || find(r->y);}
}
```

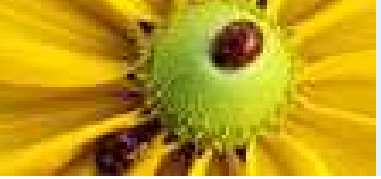


Cammino minimo

Si tratta di una classe di problemi che bisogna imparare a "vedere". Lo schema base (che presenteremo con un esempio) è il seguente: varie città sono collegate tra di loro con strade. Date due città determinare il percorso con il minor numero di chilometri.

Vi sono diversi algoritmi, il più oneroso (in termini di risorse di calcolo) ma il più compatto ed elegante richiede una funzione ricorsiva (attenzione ai cicli).

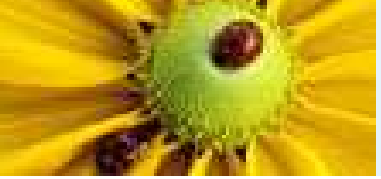




Una soluzione

Per semplicità consideriamo che le strade siano a senso unico, che vi siano al più per ogni città due strade uscenti e che non vi siano cicli. Una possibile soluzione consiste nel calcolare la distanza per tutti i possibili percorsi tra A e D e scegliere il minore. Una possibile soluzione si basa su una variante del precedente algoritmo. Quindi in pseudo codice si ha:

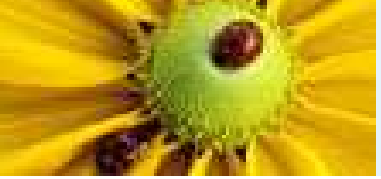
```
typedef struct node {
  char name; int xv; int yv;
  struct node *x, struct node *y }; NODE;

  void trace(NODE *r, char target, int t) {
  if (r == NULL) printf("cul de sac\n); else
  if (r->name == target) printf("%d\n",t);
  else { if (r->xv != 0) trace(r->x,target,t+xv);
            if (r->yv != 0) trace(r->y,targer,t+yv);
         }
  }
```



Pregara: linguaggio C (1) 3

Quale dei seguenti frammenti di codice produce, sulle variabili a e b, un effetto diverso dagli altri?

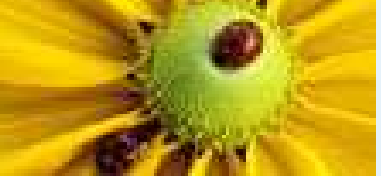
1. `int a,b; a=b;`
2. `int a,b, *p,*c; p=&a; c=&b; *p=*c;`
3. `int a,b, *p,*c; p=&a; c=&b; *p=b;`
4. `int a,b, *p,*c; p=&b; c=&a; *c=b;`
5. `int a,b, *p,*c; p=&a; c=&b; p=c;`
6. `int a,b, *p,*c; p=&a; c=p; *c=b;`



Pregara: linguaggio C (2) 1

Determinare quale è la relazione che assume valore vero quando x è esterno all'intervallo [A,B] e y è interno allo stesso intervallo?

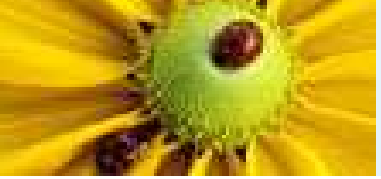
1. `(x<A) && (x>B) && (y>=A) && (y<B);`
2. `(((x<A)||(x>B)) && ((y>=A)&&(y<=B));`
3. `(((x<A)||(x>B)) && ((y>=A)||(y<=B));`
4. `(((x<A)||(x>B)) || ((y>=A)||(y<=B));`
5. `(((x<A)&&(x>B)) && ((y>=A)||(y<=B));`
6. `(((x<A)||(x>B)) || ((y>=A)&&(y<B));`



Pregara: linguaggio C (3) 2

Quale sarà il contenuto del vettore V al termine del ciclo se è V=1,21,31,4,51,6 e n=6?

```
t=n/2;

for (i=0; i<t ;i=i+1)
{
      temp=V[i];
      V[i]=V[n-i-1];
      V[n-i-1]=temp;
}
```



Pregara: linguaggio C (4) 3

Determinare T e X in modo tale che il programma renda uguali a 0 gli elementi delle due diagonali di una matrice quadrata A di n x n interi. Ad esempio:

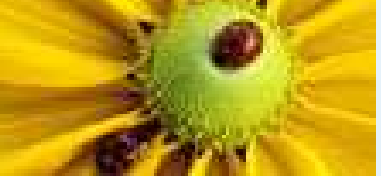
```
A={{1 2 3 4},{2 3 4 5},{3 4 5 6},{3 4 5 8}};
```

deve diventare

```
A={{0 2 3 0},{2 0 0 5},{3 0 0 6},{0 4 5 0}};
```

Soluzione

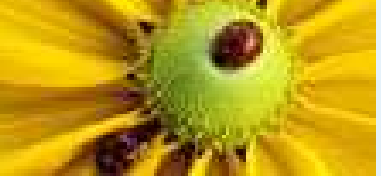
```
for (i=0; i<n ;i=i+1)
{
     A[i][T]=0;
     A[i][X]=0;
}
```



Pregara: linguaggio C (5) 4

Il codice C di seguito riportato è relativo a quello di un sottoprogramma copstr che consente di copiare una stringa t in una stringa s. Determinare gli elementi X1, X2, X3, X4 in modo tale da rendere il sottoprogramma di copia funzionante.

```
void copstr(char *s, X1)
{
        while ((*s=X2)!='X3')
        {
                X4;
                t++;
        }
}
```

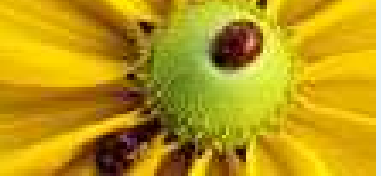


Pregara: linguaggio C (6) 2

Sia dato il seguente frammento di codice:

```
int a,b;
....
if (a>0)
if (b>0) printf("ok");
else printf("ok");
```

Discutere l'effetto dell'esecuzione di tale istruzione al variare di a e b.



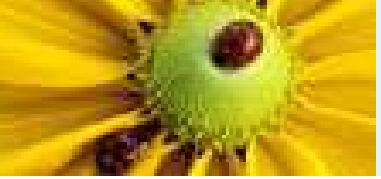
Pregara: linguaggio C (7) 1

Siano a e b due stringhe (cioè, `char a[N],b[N];`). Cosa fa il seguente frammento di programma?

```
for(i=0;a[i]!='\0';i++) b[i]=a[i];
b[i]='\0';
```

Oppure il seguente programma:

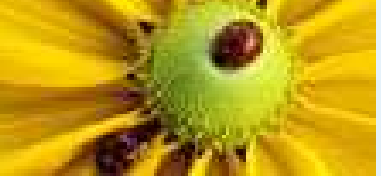
```
while(*a++=*b++); *a='\0';
```



Pregara: linguaggio C (8) 2

Qual è il ciclo corretto per trovare l'indice del primo elemento negativo in un vettore dichiarato come `float v[MAX]`

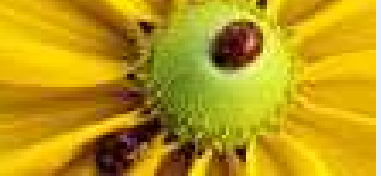
1. `while(i<MAX) { if(v[i]>=0) i++; }`
2. `while(v[i]>=0 && i<MAX) i++;`
3. `while(i<MAX || v[i]<0) i++;`
4. `while(i<MAX && v[i]>=0.0) i++;`



Pregara: linguaggio C (9) 1

La variabile dichiarata come `char d[N][M];` è ...?

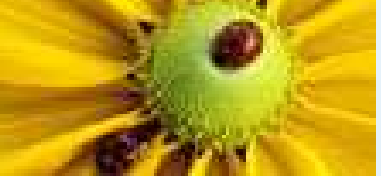
1. ...un vettore di `N` stringhe di lunghezza `M-1`
2. ...un vettore di `M` stringhe di lunghezza `N`
3. ...una matrice `NxM` di stringhe
4. ...una matrice `NxM` di stringhe lunghe un carattere ciascuna



Pregara: linguaggio C (10) 2

Cosa fa il seguente frammento di programma?

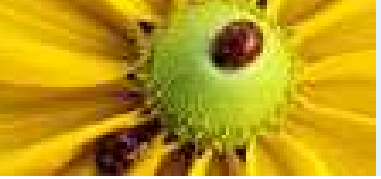
```
int i=0;
while(i<50)
if (i%2) printf("%d",i);
i++;
```



Pregara: linguaggio C (11) 1

Sia data una variabile dichiarata come `char c;`. Ipotizzando che essa contenga un carattere compreso tra `'0'` e `'9'`, come si trasferisce in una variabile intera `v` il valore decimale della cifra rappresentata da `c`?

1. `v = atoi(c);`
2. `v = (int) c ;`
3. `v = c - '0';`
4. `v = c ;`

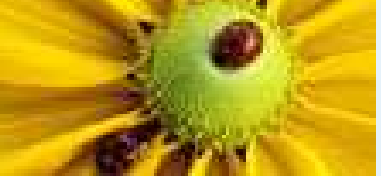


Pregara: linguaggio C (12) 3

Sia definita la seguente funzione:

```
int cond(){ return cond(); }
```

1. Cosa succede all'istruzione
   ```
   if ((a>0) && cond()) printf("ciao");
   ```
2. Cosa succede all'istruzione
   ```
   if (cond() && (a>0)) printf("ciao");
   ```

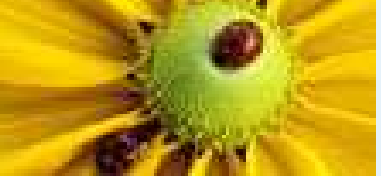


Pregara: linguaggio C (13) 2

Sia dichiarato un array

```
int a[10]
        for (i=0; i<=10; i++)
                a[i]=2;
```

1. Tutto l'array viene inizializzato al valore 2;
2. Si superano i limiti dell'array. L'errore viene segnalato in fase di compilazione;
3. Si superano i limiti dell'array. L'errore viene segnalato all'esecuzione del codice;
4. Si superano i limiti dell'array. L'errore non viene segnalato, ma si hanno conseguenze imprevedibili;
5. Si superano i limiti dell'array. All'accesso all'undicesimo elemento, il sistema operativo assegna altro spazio all'array a e l'esecuzione prosegue.



Pregara: linguaggio C (14) 3

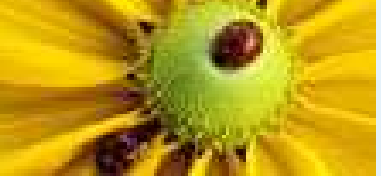
Siano dichiarate tra variabili intere a, b, c, ed un puntatore p con le dichiarazioni

```
int a, b, c;
int *p;
```

Cosa succede all'esecuzione del seguente codice:

```
for (i=0, p=&a; i<3; p++)
        *p = -1;
```

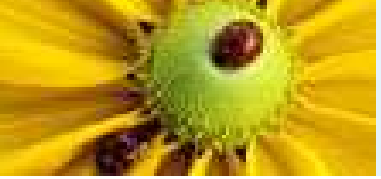
1. a, b, c vengono inizializzate al valore -1;
2. il valore -1 viene assegnato certamente ad a, e chissà a quale altra variabile in memoria, con risultati imprevedibili;
3. il compilatore segnala un errore (quale?);
4. l'esecuzione viene abortita per un errore run-time.



Pregara: linguaggio C (15) 2

Dire quale dei programmi seguenti calcola in `j` l'indice dell'elemento massimo del vettore di interi positivi v contenente n elementi, posto che `j` sia inizializzato a zero:

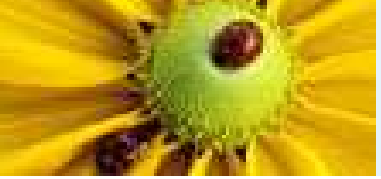
1. `for(i=0; i<n; i++) if (v[i]>j) j = v[i];`
2. `for(i=0; i<n; i++) if (v[i]>v[j]) j = i;`
3. `for(i=0; i<n; i++) if (v[j]>i) j = i;`



Pregara: linguaggio C (16) 2

Dire quale dei programmi seguenti ordina il vettore di interi v contenente n elementi:

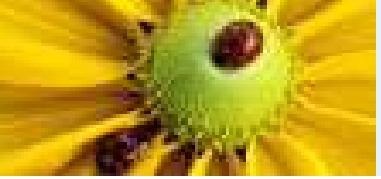
1.
```
for(i=0; i<n; i++) {
    m = i;
    for(j=i+1; j<n; j++) if (v[j]<v[m]) m=j;
    t = v[m]; v[m] = v[i]; v[i] = t;
}
```
2.
```
for(i=0; i<n; i++) {
    m = v[i];
    for(j=i; j<n; j++) if (j<m) m = j;
    t = v[m]; v[m] = v[i]; v[i] = t;
}
```
3.
```
for(i=0; i<n; i++) {
    m=i;
    for(j=i+1; j<n; j++) if (v[j]<v[m]) m=j;
    t = v[m]; v[m] = v[j]; v[j] = t;
}
```



Pregara: linguaggio C (17) 5

Considerate le seguenti definizioni dire quali delle seguenti affermazioni sono vere (è possibile che più di una affermazione sia vera: occorre indicarle tutte):

Definizioni	Affermazioni
■ un simbolo è una cifra o una lettera; ■ un codice è una sequenza (stringa) di simboli che inizia e termina con una lettera, che contiene almeno una cifra, e che non contiene mai due cifre consecutive; ■ dato un codice, un suo sottocodice è una sua sottostringa che sia a sua volta un codice; ■ un codice è minimo se non contiene sottocodici (a parte lui stesso).	1. ogni codice ha lunghezza maggiore o uguale a 3; 2. nessun codice contenente più di due lettere è minimo; 3. ogni codice ha lunghezza minore o uguale a 36; 4. il numero di sottocodici minimi di un codice dato è uguale al numero di cifre che esso contiene; 5. il numero di sottocodici minimi di un codice dato è minore o uguale alla metà del numero di lettere che esso contiene; 6. nessun codice di lunghezza maggiore di 3 è minimo.

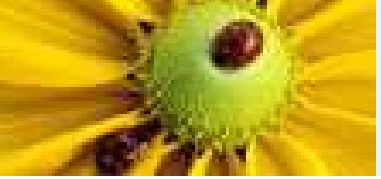


Pregara: linguaggio C (18) 5

Considerate la seguente funzione:

```
int f (int x)
{
        if (!x) return 0;
        return (x%2)+f(x/2);
}
```

Tale funzione viene eseguita su tutti i valori di x compresi fra 100 e 500. Chiamate a il minimo valore ottenuto in queste esecuzioni, e b il massimo valore; quanto valgono a e b?



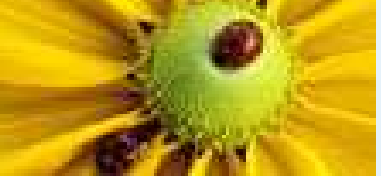
Pregara: linguaggio C (19) 3

Considerate il seguente frammento di codice:

```
float i,f;
...
f=1/50.0;
for (i=0.0; i!=1.0; i+=f) printf("A");
```

Quale dei seguenti effetti ha il ciclo for indicato?

1. Stampa 50 volte il carattere `A`;
2. Stampa 51 volte il carattere `A`;
3. Stampa 49 volte il carattere `A`;
4. Il ciclo potrebbe non terminare, stampando infinite volte il carattere `A`;
5. Il compilatore segnala un errore;
6. Il ciclo non viene mai eseguito.



Pregara: linguaggio C (20) 3

Qual è l'output del seguente programma?

```
#include <stdio.h>
#define prod(a,b) a*b

int main() {
        int a=10,b=5;
        printf("%d\n",prod(a+b,a-b));
        return 0;
}
```