1. Stabilite se ognuna delle seguenti frasi è vera o falsa.
   F a) La funzione printf comincia a stampare sempre all'inizio di una nuova riga.
   F b) I commenti fanno s. che il computer stampi sullo schermo il testo dopo // durante l'esecuzione del programma.
   V c) La sequenza di escape \n, quando . usata in una stringa di controllo per il formato in un'istruzione printf, fa posizionare il cursore all'inizio della riga successiva sullo schermo.
   V d) Prima di essere usate, tutte le variabili devono essere definite.
   V e) A tutte le variabili deve essere assegnato un tipo quando sono definite.
   F f) Il linguaggio C considera identiche le variabili number e NuMbEr.
   V g) Le definizioni possono comparire ovunque nel corpo di una funzione.
   F h) Tutti gli argomenti che seguono la stringa di controllo per il formato in una funzione printf devono essere preceduti dal simbolo &.
   V i) L'operatore di resto (%) pu. essere usato soltanto con operandi interi.
   F j) Gli operatori aritmetici *, /, %, + e – hanno tutti lo stesso livello di precedenza.
   F k) Un programma che stampa tre righe di output deve contenere tre istruzioni printf.

2. Identificate e correggete gli errori in ciascuna delle seguenti istruzioni:
   a) printf("The value is %d\n", &number);
   b) scanf("%d%d", &number1, &number2);
   c) if (c < 7) {
         printf("C is less than 7\n");
      }
   d) if (c => 7) {
         printf("C is greater than or equal to 7\n");
      }

3. Identificate e correggete gli errori in ognuna delle seguenti istruzioni:
   a) while (c <= 5) {
         product *= c;
         ++c;
      }
   b) scanf("%4f", &value);
   c) if (gender == 1)
         puts("Woman");
      else
         puts("Man");

4. Scrivete un'istruzione in C per eseguire ognuno dei seguenti compiti.
   a) Definite le variabili sum e x come tipo int. INT SUM, X;
   b) Ponete la variabile x a 1. X = 1;
   c) Ponete la variabile sum a 0. SUM = 0;
   d) Sommate la variabile x alla variabile sum e assegnate il risultato alla variabile sum. SUM += X;
   e) Stampate "The sum is: " seguito dal valore della variabile sum. PRINTF("THE SUM IS: %od", SUM);

5. Stabilite se le seguenti affermazioni sono vere o false.
   F a) Un puntatore dichiarato void pu. essere dereferenziato.
   V b) I puntatori a tipi differenti non possono essere assegnati l'uno all'altro senza un operatore cast.

   L'istruzione IF / SWITCH . usata per prendere decisioni.

6. Trovate l'errore in ognuno dei seguenti segmenti di programma. Presupponete le seguenti dichiarazioni e inizializzazioni:
   ```
   int *zPtr; // zPtr fara' riferimento all'array z
   int *aPtr = NULL;
   void *sPtr = NULL;
   int number;
   int z[5] = {1, 2, 3, 4, 5};
   sPtr = z;
   ```
   a) ++zPtr; ZPTR NON INIZIALIZZATO. ERRORE LOGICO
   b) // usa il puntatore per ottenere il primo valore dell'array;
   // si suppone che zPtr sia inizializzato
   number = *zPtr; ASSEGNAZ. INTERO CON PUNTATORE

c) // assegna l'elemento 2 dell'array (il valore 3) a number;
// si suppone che zPtr sia inizializzato
number = *zPtr[2]; DEREFERENCIANDO UN VALORE, NON UN INDIRIZZO
d) // stampa l'intero array z; si suppone che zPtr sia inizializzato
for (size_t i = 0; i <= 5; ++i) { zPtr[5] VA FUORI
    printf("%d ", zPtr[i]);
}
e) // assegna a number il valore a cui punta sPtr
number = *sPtr; NO DEREFER. CON PUNTATORE VOID
f) ++z; z = z + 1;

7. Per ognuna delle seguenti operazioni, scrivete un'istruzione che la esegua. Supponete che le variabili in virgola mobile number1 e number2 siano state definite e che number1 sia inizializzato a 7.3.
   a) Definire la variabile fPtr come puntatore a un oggetto di tipo float. FLOAT *FPTR
   b) Assegnare l'indirizzo della variabile number1 alla variabile puntatore fPtr. FPTR= &NUMBER1
   c) Stampare il valore dell'oggetto puntato da fPtr. PRINTF("%F", *FPTR)
   d) Assegnare il valore dell'oggetto puntato da fPtr alla variabile number2. NUMBER2 = *FPTR
   e) Stampare il valore di number2. PRINTF("%F", NUMBER2)
   f) Stampare l'indirizzo di number1. Usate lo specificatore di conversione %p. PRINTF("%P", &NUMBER1)
   g) Stampare l'indirizzo memorizzato in fPtr. Usare lo specificatore di conversione %p. Il valore stampato . lo stesso dell'indirizzo di number1? PRINTF("%P", FPTR). SI

8. Cos'è sbagliato nella seguente istruzione di iterazione while (supponete che z abbia il valore 100) che deve calcolare la somma dei numeri interi da 100 in gi. fino a 1?
```
while ( z >= 0 )
    sum += z;
```
LOOP

9. Trovate l'errore in ognuno dei seguenti segmenti di codice e spiegate come correggerlo.
   a)
```
x = 1;
while (x <= 10);
    ++x;
```
OPPURE
```
x = 1
WHILE (x<=10){
    ++x;
}
```
   b)
```
for (double y = .1; y != 1.0; y += .1) {
    printf("%f\n", y);
}
```
```
FOR(INT y = 1; y != 10; ++y){
    PRINTF("%F", (FLOAT) y /10),
}
```
   c)
```
switch (n) {
    case 1:
        puts("The number is 1"); BREAK;
    case 2:
        puts("The number is 2");
        break;
    default:
        puts("The number is not 1 or 2");
        break;
    }
```

10. Per il seguente programma determinate l'estensione del campo d'azione (alla funzione, al file, al blocco o al prototipo di funzione) di ognuno dei seguenti elementi.
   a) La variabile x in main. BLOCCO
   b) La variabile y in cube. BLOCCO
   c) La funzione cube. FILE
   d) La funzione main. FILE
   e) Il prototipo di funzione per cube. FILE
   f) L'identificatore y nel prototipo di funzione per cube. PROTOTIPO DI FUNZIONE

```
1 #include <stdio.h>
2 int cube(int y);
3
4 int main(void)
5 {
6     for (int x = 1; x <= 10; ++x)
7         printf("%u\n", cube(x));
8 }
9
10 int cube(int y)
```

```
11 {
12         return y * y * y;
13 }
```

11. Scrivete l'intestazione di funzione per ognuna delle seguenti funzioni.
    a) La funzione hypotenuse che riceve due argomenti in virgola mobile in doppia precisione, side1 e side2, e restituisce un risultato in virgola mobile in doppia precisione. `DOUBLE FUN(DOUBLE A,DOUBLE B)`
    b) La funzione smallest che riceve tre interi, x, y, z, e restituisce un intero. `INT SMALLEST(INT X, INT Y, INT Z)`
    c) La funzione instructions che non riceve alcun argomento e non restituisce alcun valore. [Nota: tali funzioni sono comunemente usate per dare delle istruzioni a un utente.] `VOID INSTRUCTIONS()`
    d) La funzione intToFloat che riceve un argomento intero, number, e restituisce un risultato in virgola mobile. `FLOAT INTTOFLOAT(INT NUMBER)`

12. Scrivete istruzioni per effettuare le seguenti operazioni:
    a) Definite table come un array intero con 3 righe e 3 colonne. Supponete che la costante simbolica SIZE sia stata definita come valore 3. `INT TABLE[SIZE][SIZE]`
    b) Quanti elementi contiene l'array table? Stampate il numero totale di elementi. `9. PRINTF("%d",SIZE*SIZE)`
    c) Usate un'istruzione di iterazione for per inizializzare ogni elemento di table alla somma dei suoi indici. Supponete che le variabili intere x e y siano definite come variabili di controllo. `FOR(INT X=0, X<SIZE, X++){ FOR(INT Y=0, Y<SIZE, Y++) TABLE[X][Y]=X+Y;`
    d) Stampate i valori di ogni elemento dell'array table. Supponete che l'array sia stato inizializzato con la definizione: `FOR(INT X=0, X<SIZE, X++){ FOR(INT Y=0, Y<SIZE, Y++) PRINTF("TABLE[%d][%d]=%d",X,Y,TABLE[X][Y]);}`
    int table[SIZE][SIZE] ={ { 1, 8 }, { 2, 4, 6 }, { 5 } };

13. Scrivete un'istruzione o un insieme di istruzioni per eseguire ognuno dei seguenti compiti.
    a) Sommate i numeri interi dispari tra 1 e 99 usando un'istruzione for. Utilizzate variabili intere sum e count senza segno. `FOR(INT I=1; I<100; I++){UNSIGNED COUNT=1; IF(COUNT%2==1) UNSIGNED SUM+=COUNT;}`
    b) Stampate il valore 333.546372 con un'ampiezza di campo di 15 caratteri con precisioni di 1,2,3,4 e 5. Allineate a sinistra l'output. Quali sono i cinque valori che vengono stampati? `PRINTF("%15.1F",337.546372); PRINTF("%15.2F",337.546372);`
    c) Calcolate il valore di 2.5 elevato alla potenza di 3 usando la funzione pow. Stampate il risultato con una precisione di 2 con una larghezza di campo di 10 posizioni. Qual . il valore che viene stampato? `R=POW(2.5,3); PRINTF("%10.2F",R);`
    d) Stampate gli interi da 1 a 20 usando un ciclo while e la variabile contatore x. Stampate soltanto cinque interi per riga. [Suggerimento: usate il calcolo x % 5. Quando il valore di questo . 0, stampate un carattere newline, altrimenti stampate un carattere tab.]
    ```
    INT X=1;
    WHILE(X<=20){
        PRINTF("%d",X);
        IF(X%5==0) PUTS("\n");
        ELSE PUTS("\t");}
    ```

14. Scrivete una singola istruzione per eseguire ognuna delle seguenti operazioni. Supponete che le variabili c (che memorizza un carattere), x, y e z siano di tipo int, che le variabili d, e e f siano di tipo double, che la variabile ptr sia di tipo char * e che gli array s1[100] e s2[100] siano di tipo char.
    a) Convertite il carattere memorizzato nella variabile c in una lettera maiuscola. Assegnate il risultato alla variabile c. `C=TOUPPER(C);`
    b) Determinate se il valore della variabile c . una cifra. Usate l'operatore condizionale come mostrato nelle Figure 8.2–8.4 per stampare " is a " o " is not a " quando il risultato viene stampato. `PRINTF("'%c'%s DIGIT ",C,ISDIGIT(C)?"IS A":"IS NOT A");`
    c) Determinate se il valore della variabile c . un carattere di controllo. Usate l'operatore condizionale per stampare " is a " o " is not a " quando viene stampato il risultato. `PRINTF("'%c'%s CONTROL CH.",C,ISCNTRL(C)?"IS A":"IS NOT A");`
    d) Leggete una riga di testo dalla tastiera e memorizzatelo nell'array s1. Non usate scanf. `FGETS(S1,100,STDIN)`
    e) Stampate la riga di testo memorizzata nell'array s1. Non usate printf. `PUTS(S1);`
    f) Assegnate a ptr l'indirizzo dell'ultima occorrenza di c in s1. `PTR=STRRCHR(S1,C);`
    g) Stampate il valore della variabile c. Non usare printf. `PUTCHAR(C);`
    h) Determinate se il valore di c . una lettera. Usate l'operatore condizionale per stampare " is a " o " is not a " quando viene stampato il risultato. `PRINTF("'%c'%s LETTER",C,ISALPHA(C)?"IS A":"IS NOT A");`
    i) Leggete un carattere dalla tastiera e memorizzatelo nella variabile c. `C=GETCHAR();`
    j) Assegnate a ptr la locazione della prima occorrenza di s2 in s1. `PTR=STRSTR(S1,S2);`
    k) Determinate se il valore della variabile c . un carattere stampabile. Usate l'operatore condizionale per stampare " is a " o " is not a " quando viene stampato il risultato. `PRINTF("'%c'%s PRINTING CH.",C,ISPRINT(C)?"IS A":"IS NOT A");`
    l) Leggete tre valori double dalla stringa "1.27 10.3 9.432" e memorizzateli nelle variabili d, e e f. `SSCANF("1.27 10.3 9.432","%F%F%F",&D,&E,&F);`
    m) Copiate la stringa memorizzata nell'array s2 nell'array s1. `STRCPY(S1,S2);`
    n) Assegnate a ptr la locazione della prima occorrenza di un carattere di s2 in s1. `PTR=STRPBRK(S1,S2);`
    o) Confrontate la stringa in s1 con la stringa in s2. Stampate il risultato.
    `PRINTF("STRCMP(S1,S2)=%d",STRCMP(S1,S2));`

p) Assegnate a ptr la locazione della prima occorrenza di c in s1. PTR = STRCHR (S1, C)
q) Usate sprintf per memorizzare la stampa dei valori delle variabili intere x, y e z nell'array s1. Ogni valore va stampato con un'ampiezza di campo uguale a 7. SPRINTF (S1, "%7d%7d%7d", x, y, z);
r) Aggiungete 10 caratteri della stringa in s2 in coda alla stringa in s1. STRNCAT (S1, S2, 10);
s) Determinate la lunghezza della stringa in s1. Stampate il risultato. PRINTF("%u", STRLEN(S1));
t) Assegnate a `ptr` la locazione del primo token in `s2`. I token nella stringa `s2` sono separati da virgole (`,`). PTR = STRTOK (S2, ",");

15. I programmi che traducono nel linguaggio macchina programmi in un linguaggio di alto livello sono chiamati COMPILATORI.

16. Analizzate il significato di ognuno dei seguenti nomi:
a) stdin STANDARD INPUT
b) stdout STANDARD OUTPUT
c) stderr STANDARD ERROR
FLUSSI STANDARD PER INPUT, OUTPUT E OUTPUT DEGLI ERRORI

17. Ogni programma in C inizia l'esecuzione dalla funzione MAIN().

18. Stabilite se le seguenti affermazioni sono vere o false: se la risposte
a) Il caso default . necessario nell'istruzione di selezione switch. F. NON È OBBLIGATORIO
b) L'istruzione break . necessaria nel caso default di un'istruzione di selezione switch. F. BREAK NON È NECESSARIO IN NESSUN CASO
c) L'espressione (x > y && a < b) . vera se x > y . vera o se a < b . vera. F. ENTRAMBE DEVONO ESSERE VERE
d) Un'espressione contenente l'operatore || . vera se uno o ambedue gli operandi sono veri. V

19. Trovate l'errore in ognuno dei seguenti segmenti di programma e spiegate come correggerlo:
a) char s[10];
strncpy(s, "hello", 5); STRNCPY (S, "HELLO", 6)
printf("%s\n", s);
b) printf("%s", 'a'); PRINTF("%C", 'A') OPPURE PRINTF("%S", "A")
c) char s[12]; S[13] //PIÙ ELEMENTI. CARATT. FINE STRINGA
strcpy(s, "Welcome Home");
d) if (strcmp(string1, string2)) {
puts("The strings are equal");
}
STRCMP TORNA 0 SE LE STRINGHE SONO UGUALI. ==0

20. Mostrate due metodi differenti per inizializzare l'array di caratteri vowel con la stringa di vocali "AEIOU".

1) CHAR VOWEL[] = {'A', 'E', 'I', 'O', 'U', '\0'};
2) CHAR VOWEL[] = "AEIOU";