Carrazza	☐
Mereghetti	☐
Spoletini	☐
Tamascelli	☐

INFORMATICA – Preparazione scritto – 21 gennaio 2019

Cognome _____ Nome _____ Matr _____

Lab (voto/quando) _____/_____ Firma _____

1) Siano date le seguenti definizioni:

```
struct elem {
      int item;
      int *p;
};

elem *X = new elem[100]; elem *z; int y;
```

- Come assegnare al campo **item** dell'elemento di **X** di indice **3** la somma del campo **item** dell'ultimo elemento di **X** e del contenuto della cella puntata dal campo **p** del primo elemento di **X**?

```
X[3].item = X[99].item + *(X[0].p)
(X+3) -> item = (X+99) -> item + *(X -> p)
```

- Come assegnare l'indirizzo di **y** al campo **p** del primo elemento di **X**?

```
X[0].p = &y
X -> p = &y
```

- Come creare dinamicamente una variabile intera e farla puntare dal campo **p** dell'elemento puntato da **z**?

```
*(z).p = new int
z -> p = new int
```

- Assumendo che **X** valga **45** e che un puntatore ad **int** occupi **2 byte**, quanto vale **X+10** ?

105

2) Quale sarà il valore finale della variabile **a** dopo l'esecuzione del seguente frammento di codice?

```
int A[] = {1, 2, 3, 4, 5, 6, 7}; int a = 0;

for ( int i = 0; i < 6; i += 2 ) {
      ++i;
      a = a + A[i]/(float)i;
}
```

a = 3

3) Inizializzate la variabile **matr** col vostro numero di matricola (per questa esercitazione inizializzeremo **matr** con **305011**) e dite quanto valgono le variabili **a**, **c** e **x** al termine dell'esecuzione del codice seguente:

```
int matr = 305011; float a = matr;
int c = (matr/1000)*1000 + matr%1000;
a = (a/1000)*1000 + c;
float x = (int)matr/2.0 + (int)(matr/2.0)
```

a = 610022.0 c = 305011 x = 305010.5

4) La struttura

```
struct cerchio {
      float x, y, r;
}
```

rappresenta un cerchio nel piano cartesiano avente centro nel punto di coordinate **(x,y)** e raggio **r**. Scrivete un frammento di codice che utilizzi la struttura **cerchio** per memorizzare un insieme di cerchi la cui numerosità è preventivamente chiesta all'utente. Successivamente, l'utente inserisce i cerchi ed il codice deve *impedire la memorizzazione di cerchi con raggio nullo o negativo*. Al termine, il codice deve stampare i soli cerchi che *giacciono completamente nel primo quadrante*.

```
int dim;
cout << "Quanti cerchi: ";
cin >> dim;
cerchio *C = new cerchio[dim];
for ( int i = 0; i < dim; i++ ) {
      cin >> C[i].x >> C[i].y >> C[i].r;
      while ( C[i].r <= 0 ) {
            cout << "Reinserire il raggio: ";
            cin >> C[i].r;
      }
}
for ( int i = 0; i < dim; i++ )
      if ( C[i].x > C[i].r && C[i].y > C[i].r ) {
            cout << "x = " << C[i].x << " y = " << C[i].y << " r = " << C[i].r;
            cout << endl;
      }
```

5) Scrivete la funzione

bool consEven(int *X, int dim, int p)

che accetta in ingresso un array di interi positivi **X** di dimensione **dim** e un intero positivo **p**. La funzione deve restituire **true** se **X** contiene una sequenza di *almeno* **p** *numeri pari consecutivi*, **false** altrimenti.

Esempio: assumendo l'array **X = {5, 6, 28, 4, 13, 13}**, la chiamata **consEven(X, 6, 2)** deve restituire **true**.

```
bool consEven(int *X, int dim, int p) {
      int k = 0;
      for ( int i = 0; i < dim; i++ )
            if ( X[i]%2 == 0 ) {
                  k++;
                  if ( k >= p )
                        return true;
            } else
                  k = 0;
      return false;
}
```

5) Scrivete la funzione **sortEven** che accetta in ingresso un array di interi positivi di dimensione **dim** e ordina in senso crescente *solamente gli interi pari* contenuti nell'array, lasciando al loro posto gli altri.

Esempio: assumendo l'array **X = {5, 6, 28, 4, 13, 2}**, la chiamata **sortEven(X, 6)** deve modificare **X** come **X = {5, 2, 4, 6, 13, 28}.**

```
void sortEven(int *X, int dim) {
      for ( int i = 0; i < dim-1; i++ )
            if ( X[i]%2 == 0 )
                  for ( int j = i+1; j < dim; j++ )
                        if ( X[j]%2 == 0 && X[j] < X[i] )
                              swap( X[i], X[j] );
}

void swap(int &x, int &y) {
      int t = x;
      x = y;
      y = t;
}
```

6) Il vettore **int V[100]** è caricato nelle sue prime **50** componenti da interi positivi. Scrivete un frammento di codice che operando solo su **V**, con l'uso eventuale di poche variabili ausiliarie *ma senza usare un vettore ausiliario*, crei una copia consecutiva di ogni intero pari contenuto in **V** ed elimini ogni intero dispari. Al termine, il codice deve anche stampare quanti pari sono stati aggiunti e quanti dispari eliminati.

Esempio: **V** iniziale: **4 7 7 12 9 15 18 18 32 ...**
V finale: **4 4 12 12 18 18 18 18 32 32 ...**

```
int d = 50, pr = 0, dr = 0;
for ( int i = 0; i < d; i++ )
      if ( X[i] % 2 == 0 ) {
            for ( int j = 98; j >= i; j-- )
                  X[j+1] = X[j];
            d++;
            i++;
            pr++;
      }
      else {
            for ( int j = i+1; j < 100; j++ )
                  X[j-1] = X[j];
            d--;
            i--;
            dr++;
      }

cout << "Pari aggiunti: " << pr << " Dispari tolti: " << dr << endl;
```

7) Sia **float P[10][2]** un insieme di **10** punti nel piano cartesiano, ove le coordinate del punto **i**-esimo sono rispettivamente **P[i][0]** e **P[i][1].** Scrivete un frammento di codice che individui la *minima* distanza tra le coppie di punti in **P.**

```
float d = dist( P, 0, 1 );
for ( int i = 0; i < 9; i++ )
      for ( int j = i+1; j < 10; j++ )
            if ( dist(P, i, j) < d )
                  d = dist(P, i, j);
cout << "Distanza minima: " << d << endl;

float dist(float X[][], int a, int b) {
      return sqrt( pow(X[a][0]-X[b][0],2) + pow(X[a][1]-X[b][1],2) );
}
```