

## Esercizio 1

Data la seguente porzione di programma rispondere, dando motivazione, alle domande corrispondenti:

```
int f (int& k) {return k+2}

int main() {
    int* matricola = new int[6] { //Inserire qui la tua matricola};

    // 1. La seguente istruzione è corretta? Se sì, cosa stampa?
    cout << *(matricola[2]) << endl;

    // 2. La seguente istruzione è corretta? Se sì, che cosa stampa?
    matricola[3] = f(matricola[2]);
    cout << matricola[3] << " " << matricola[2] << endl;

    // 3. Le seguenti istruzioni sono corrette? Motivare la risposta
    float* m = new float(6);
    for (int i=0; i < 6; i+=2) {
        m+=matricola[i] * matricola[i+1];
    }
    cout << m/=4 << endl;

    // 4. Scrivere sul foglio le istruzioni per deallocare la memoria
    dinamica allocata nel programma.

    return 0;
}
```

## Esercizio 2

Si richiede di implementare la classe `VettorePazzo`, ereditando opportunamente da `vector<unsigned>`, che memorizzi numeri interi positivi e un `numero_pazzo` ricevuto a tempo di costruzione. Deve permettere almeno l'inserimento in coda e l'accesso per indice tramite operatore `[]`, allo stesso modo di come avviene per un classico `vector`, rispettando inoltre le seguenti condizioni:

- Se si prova ad inserire il `numero_pazzo` o un suo multiplo, nel vettore non viene inserito nulla;
- Se il vettore contiene un numero di elementi multiplo del `numero_pazzo`, l'operatore `[]` restituirà sempre 0;
- L'operatore `[]`, inoltre, non deve mai andare in errore. Sia  $n$  il numero di elementi attualmente presenti nel vettore e  $i$  l'indice a cui si sta provando ad accedere. L'operatore deve essere implementato secondo il seguente criterio:
  - Se  $i < 0$  e  $|i| \geq n$ , restituisce sempre il primo elemento del vettore;
  - Se  $i < 0$  e  $|i| < n$ , restituisce l' $i$ -esimo elemento da destra;
  - Se  $i \geq 0$  e  $|i| < n$ , restituisce l' $i$ -esimo elemento da sinistra;
  - Se  $i \geq 0$  e  $|i| \geq n$ , restituisce sempre l'ultimo elemento del vettore.
- Due `VettorePazzo`, se confrontati tra di loro, risultano sempre uguali, mentre, per stabilire se un `VettorePazzo` è minore di un altro, si prende in considerazione soltanto il loro `numero_pazzo`.

