1 Question 1

1.1 a

See code.

1.2 b

For different matrix sizes we want to test the following properties:

- Setting/Getting: $A(i,j) = a \Rightarrow A(i,j)$ returns a
- Symmetry: $A(i,j) == A(j,i)$ is true $\forall(i,j)$.
- l_0 norm: returns correct number of nonzeros in whole matrix A.

Given matrix A, with dimensions $N \times N$, we implement the following test.

$\forall(i,j)$ with $j \geq i$ (upper triangular part, i,j zero-indexed), assign $A(i,j) = i+j$.

Verify following statements

- $i,j = 0,...,n-1 \Rightarrow A(i,j) = i+j$
- $i,j = 0,...,n-1 \Rightarrow A(i,j) == A(j,i)$ is true.
- $A.l_0\_norm()$ returns n^2-1.

1.3 c

The submitted code passes all tests.

For example, setting $N=5$ the first two conditions pass and $A.l_0\_norm()$ returns 24.

2 Question 2

See code.

3 Question 3

See code.

4 Question 4

4.1 a

For the non-void daxpy, we can create a new vector and use a **std::for_each** to push the transformed elements into the new vector without altering in the input vector.

For the void daxpy, we can overwrite the current vector in place using **std::transform**.

Using the test $a = 2$, $y = 3$ on the input vector $[1, 2, 3]$ we get the output vector $[5, 7, 9]$ in both cases.

4.2 b

For a single student we can use a lambda function to compute the weighted grade for the class and return true if the student passed the class, false if not.

The function **std::for_all** can then be used with the above lambda function to determine whether a vector of students all passed the class.

Testing on students with grades $(80, 80, 80)$ and $(90, 90, 90)$ we get true. Adding the student $(10, 20, 90)$ we then get false, as expected.

4.3 c

We can first sort the entire list.

Then using two **std::for_each** calls we can select the odd and even numbers, which will still be sorted.

We can then simply use two more **std::for_each** calls to concatenate the vectors of odd and even numbers.

Testing on the vector $[4, 5, 3, 2]$ we get $[3, 5, 2, 4]$ as expected.

4.4 d

We can use the linked lists sort function along with a lambda function to encode the requirements.

For two entries, x and y, These can be written as:

- $x.row > y.row$? return false.
- $x.row < y.row$? return true.
- $x.col > y.col$? return false.

- return true otherwise since x and y have the same row but x has a smaller column

Testing on the vector $[(1,1),(0,2),(0,1),(0,0)]$ we get $[(0,0),(0,1),(0,2),(1,1)]$ as expected.