Q1

1. We could tell any difference from the screenshot below of float and double calculation

```
x: 2.3230123519897461; float: 0.73017650842666626; double: 0.7301765448678484
x: 3.7168369293212891; float: -0.54403942823410034; double: -0.5440397687696610
x: 2.4085452556610107; float: 0.66913741827011108; double: 0.6691373779344768
x: 3.8730659484863281; float: -0.66796690225601196; double: -0.6679667724977640
```

I highlight the difference of digits using the red box. The screenshot below is the precise result of the calculator. We could conclude that the double result is more precise than the float result. The reason is obviously the preciseness of the float and double of IEEE754 format. We know that float is of **32-bit** base-2 single precision and double is of **64-bit** base-2 single precision. So that's why the result of double calculation is more precise.

2. We run the calculation with and without the AVX. We generate random float and double x and calculate the value of sine for 40,000 times to calculate the running time. The result is shown below.

```
[zhang.yam@login-01 ~]$ g++ -std=c++11 Q1.cpp -o Q1 && ./Q1
Running time: 1601.11 ms
x: 2.8693957328796387; float: 0.26884832978248596; double: 0.26884811954357457
```

```
[zhang.yam@d0005 ~]$ module add gcc/10.1.0
[zhang.yam@d0005 ~]$ g++ --version
g++ (GCC) 10.1.0
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[zhang.yam@d0005 ~]$ g++ -std=c++11 -march=native Q1.cpp -o Q1 && ./Q1
Running time: 362.825 ms
x: 4.1093220710754395; float: -0.82360070943832397; double: -0.82360002884375727
```

We could tell that the running time of using AVX has improved by nearly 77%.

3. We use another method to calculate the sine. Here we use the Padé approximation. As an approximation, sine could be calculated by the below formula.

$$\sin(x) \approx \frac{(12671/4363920)x^5 - (2363/18183)x^3 + x}{1 + (445/12122)x^2 + (601/872784)x^4 + (121/16662240)x^6}$$

Here is the result of running that in the cpp program.

```
[zhang.yam@c0317 ~]$ g++ -std=c++11 Q1_2.cpp -o Q && ./Q
Running time: 0.0281029 ms
x: 2.3863945007324219; float: 0.68545013666152954; double: 0.68545015963332734
```

We could tell the consciousness is worse than the Taylor series because in the Taylor series we use more than 30 terms in our program. Although the above formula is just an approximation, we could also tell the preciseness difference between float and double data type.

References:

1. https://math.stackexchange.com/questions/2196371/how-to-approximate-sinx-using-pad%C3%A9-approximation