Implement the constructor of a C++ thread managing class and a program that tests the creation of an object of your class.

The constructor MUST create only one new thread and MUST return only after the thread begins execution.

The constructor MUST accept (1) a pointer to the entry function of the new thread and (2) multiple arguments that can be passed to the entry function.

Use any POSIX compliant thread library like pthread to implement the underlying thread mechanisms.

Your solution must compile and execute the test program successfully.

If you are unclear about anything, feel free to make assumptions and note your assumptions as comments in your solution.

The following is an incomplete illustration of how the Thread class you implement may be used and is meant to serve only as an example to help you design your solution and test program.

main.cc

-------

#include <iostream>

#include "Thread.h"

class ThreadTest

{

public:

ThreadTest() : testCount(0) { }

run(/\* args \*/)

{

std::cout << "Parent thread ID: " << pthread\_self() << std::endl;

Thread child(ThreadTest::testEntry, /\* args \*/);

child.join();

ASSERT(testCount == 1);

}

private:

int testCount;

static void testEntry(void \*v)

{

/\* call testRun() \*/

}

void testRun()

{

std::cout << "Child thread ID: " << pthread\_self() << std::endl;

++testCount;

}

};

int main()

{

Threadtest test;

test.run(/\* args \*/);

return 0;

}

Thread.h

--------

class Thread

{

public:

/\* typedef threadEntry \*/

/\* typedef threadArg \*/

Thread(threadEntry entry, threadArg arg);

~Thread();

void join(); /\* optional: implement only if you have time \*/

private:

pthread\_t tid;

};

Thread.cc

---------

/\* Implement constructor and other infrastructure \*/