1. *Create a thread in either a C or a Java program and run it. The thread should read a keyboard value “x” and exit from the code, or it can also use an “ESC” to exit. That is, the thread should run until the keyboard exit character is entered. Print your source code and output to a single file and generate a pdf and submit it in Blackboard. Put your compile and run options if any in the code as comments.*
   1. **thread\_create.c**

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

// Compile: gcc -o thread\_create thread\_create.c -lpthread

void \*read\_keys(void \* ptr);

int main()

{

pthread\_t thread1;

int iret1 = pthread\_create(&thread1, NULL, read\_keys, NULL);

if(0 != iret1)

{

printf("Error - pthread\_create() return code: %d\n", iret1);

return (-1);

}

pthread\_join(thread1, NULL);

printf("All threads complete\n");

return 0;

}

void \*read\_keys(void \*ptr)

{

printf("Enter x or X or ESC to exit\n");

int cur\_char;

do

{

cur\_char = getchar();

}

while(('x' != (char)(cur\_char)) && ('X' != (char)(cur\_char)) && (27 != cur\_char));

return 0;

}

* 1. **output**
     1. **Exit using ‘x’:**

[mjs@localhost Homework4]$ ./thread\_create

Enter x or X or ESC to exit

i

P

4

x

All threads complete

* + 1. **Exit using ‘X’:**

[mjs@localhost Homework4]$ ./thread\_create

Enter x or X or ESC to exit

8

d

S

X

All threads complete

* + 1. **Exit using ESC:**

[mjs@localhost Homework4]$ ./thread\_create

Enter x or X or ESC to exit

l

F

3

\*

^[ (ESC)

All threads complete

1. *Using the above thread program, now create 20 threads in your program and run them. No need to wait for the keyboard in each thread, just print “Hello” and its thread id in each thread on a single line and exit. Append your source code and output to the same file as listed above. Add enough comments in your program to make it readable.*
   1. **multi\_thread\_create.c**

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

// Compile: gcc -o multi\_thread\_create multi\_thread\_create.c -lpthread

void \*print\_hello(void \* ptr);

int main()

{

printf("Main thread: %u\n", pthread\_self());

pthread\_t tID[20];

// Create 20 threads

unsigned int i = 0;

for(i = 0; i < 20; ++i)

{

// Create thread

int iret1 = pthread\_create(&tID[i], NULL, print\_hello, NULL);

if(0 != iret1)

{

printf("Error - pthread\_create() return code: %d\n", iret1);

return (-1);

}

}

// Wait for all threads to complete

for(i = 0; i < 20; ++i)

{

pthread\_join(tID[i], NULL);

}

printf("All threads complete\n");

return 0;

}

void \*print\_hello(void \*ptr)

{

printf("Hello from thread %u\n", pthread\_self());

return 0;

}

* 1. **output**

[mjs@localhost Homework4]$ ./multi\_thread\_create

Main thread: 2243671872

Hello from thread 2185111296

Hello from thread 2193504000

Hello from thread 2176718592

Hello from thread 2168325888

Hello from thread 2201896704

Hello from thread 2143147776

Hello from thread 2227074816

Hello from thread 2159933184

Hello from thread 2235467520

Hello from thread 2151540480

Hello from thread 2218682112

Hello from thread 2210289408

Hello from thread 2134755072

Hello from thread 2126362368

Hello from thread 2092791552

Hello from thread 2109576960

Hello from thread 2117969664

Hello from thread 2076006144

Hello from thread 2101184256

Hello from thread 2084398848

All threads complete