```
#include
#include
using namespace std;
DWORD WINAPI ThreadFun(LPVOID lpParam){
cout << "Thread is Running" << endl;
return 0;
}
int main(){
HANDLE hThread;
DWORD ThreadId;
hThread = CreateThread(NULL, 0, ThreadFun, NULL, 0, &ThreadId);
if (hThread == NULL)
cout << "Thread Creation Fails" << GetLastError() << endl;
else{
cout << "Thread Creation Success" << endl;
cout << "Thread Id" << ThreadId << endl;
CloseHandle(hThread);
}
return 0;
}
```

The CreateThread function creates a new thread for a process. The creating thread must specify the starting address of the code that the new thread is to execute. Typically, the starting address is the name of a function defined in the program code. This function takes a single parameter and returns a DWORD value. A process can have multiple threads simultaneously executing the same function. The calling thread uses the WaitForMultipleObjects function to persist until all worker threads have terminated. The calling thread blocks while it is waiting; to continue processing, a calling thread

would use WaitForSingleObject and wait for each worker thread to signal its wait object.

To create a thread, the Windows API supplies the CreateThread( ) function. Its prototype is shown here:

```
HANDLE CreateThread(LPSECURITY_ATTRIBUTES secAttr,
SIZE_T stackSize,
LPTHREAD_START_ROUTINE threadFunc,
LPVOID param,
DWORD flags,
LPDWORD threadID);
```

There are lots of other functionalities provided by win32 API for thread management including e.g. SuspendThread and ResumeThread.

Note: You can also use the _beginthread function to create a thread. It has the following syntax:

```
uintptr_t _beginthread(void( __cdecl *start_address )( void * ),
unsigned stack_size, void *arglist );
```

The first argument is the function name, the second is the stack size (use 0 to force windows to select an appropriate size for you) and the final argument is a list of arguments that you can pass to the called function.
e.g. _beginthread(functionName, 0, (void*)12);
But to use _beginthread, you will have to change VC settings. You will have to go to Project->settings->Category->Code Generation ->Use Run Time Library->Debug Multithreaded.
11
Exercise
Q1. Implement the code given above and show the output.
Q2. Modify the program so that we have only two threads created. The first should call print_hello1 and the second should call print_hello2. Both functions should have for loops for 100 times and should display line “Executing thread no. 1” or “Executing thread no. 2”. Is the output interleaved or not. Why?
Q3. Use two threads to show two rotating bars at the two top corners of a screen. Basically each thread will be driving the “rotation” of the bar. (Hint: You can use “\” and “/” to achieve the effect but you will have to figure out how to achieve the effect of “rotation”.)