National University of Technology

Computer Science Department

Semester fall – 2025

Program: Artificial intelligence

Course: Operating System

Submitted By

Muhammad Anas

F23607044

Program Question: Implementing Threads with Join and Detach

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

// Function for first two joinable threads
void* joinable_thread(void* arg)
{
   int thread_no = *(int*)arg;
   int sleep_time = (rand() % 3) + 1;  // Sleep between 1 and 3 seconds

   printf("Thread %d: Started. Sleeping for %d seconds.\n", thread_no,
sleep_time);

   sleep(sleep_time); // Simulate time-consuming task

   printf("Thread %d: Finished.\n", thread_no);

   return NULL;
}

// Function for daemon detached thread
void* daemon_thread(void* arg)
{
   int sleep_time = (rand() % 3) + 1;

   printf("Daemon Thread: Started. Sleeping for %d seconds.\n", sleep_time);

   sleep(sleep_time); // Simulate task

   printf("Daemon Thread: Finished independently.\n");

   return NULL; }
```

```
int main()
{
   pthread_t t1, t2, t3;
   int id1 = 1, id2 = 2;

   // Initialize random number generator
   srand(time(NULL));

   // Create first two joinable threads
   pthread_create(&t1, NULL, joinable_thread, &id1);
   pthread_create(&t2, NULL, joinable_thread, &id2);

   // Create third thread and make it daemon detached
   pthread_create(&t3, NULL, daemon_thread, NULL);
   pthread_detach(t3);

   // Main thread waits for first joinable thread
   printf("Main thread: Waiting for Thread 1 to finish.\n");
   pthread_join(t1, NULL);

   // Main thread waits for second joinable thread
   printf("Main thread: Waiting for Thread 2 to finish.\n");
   pthread_join(t2, NULL);

   // Main thread exits without waiting for daemon thread
   printf("Main thread: Exiting program.\n");

   return 0;
}
```