University of Wolverhampton
School of Mathematics and Computer Science

Student Number:
Name:

6CS005 High Performance Computing Week 4 Workshop

Revision on Multithreading

Tasks – Multithreading

You will need to refer to the Week 4 lecture slides in order to complete these tasks.

1. Write a multithreaded C program to print out all the prime numbers between 1 to 10000. Use exactly 3 threads.
2. Convert this program to prompt the user for a number and then to create the number of threads the user has specified to find the prime numbers.
3. Convert the program in (2) so that each thread returns the number of prime numbers that it has found using pthread_exit() and for main program to print out the number of prime number that each thread has found.
4. Convert the program in (3) to use pthread_cancel() to cancel all threads as soon as the 5th prime number has been found.
5. In a banking system, multiple users can access and modify their accounts concurrently. Each user has a balance, and they can deposit or withdraw money. Modify the code below to use a **mutex** to prevent race conditions during balance updates and ensure that multiple users can't simultaneously update the balance of the same account.

```
typedef struct {
    int accountNumber;
    double balance;
} Account;

Account accounts[10];

void *withdraw(void *p) {
    int accountId = *(int *)p;
    double amount = 100;  // Withdraw amount
    accounts[accountId].balance -= amount;
}

void *deposit(void *p) {
    int accountId = *(int *)p;
    double amount = 100;  // Deposit amount
    accounts[accountId].balance += amount;
}

int main() {
    pthread_t threads[20];
    int ids[10];
    // Create multiple threads to simulate transactions on the same
account
    for (int i = 0; i < 10; i++) {
        ids[i] = i;
```

```
            pthread_create(&threads[i], NULL, withdraw, &ids[i]);
            pthread_create(&threads[i + 10], NULL, deposit, &ids[i]);
        }
        for (int i = 0; i < 20; i++) {
            pthread_join(threads[i], NULL);
        }
        // Print final balance
        for (int i = 0; i < 10; i++) {
            printf("Account %d balance = %.2f\n", i,
accounts[i].balance);
        }
}
```

6. A printer is shared among multiple users. Each user can either print a document or wait if the printer is in use. There are only 2 printers in the office. Use **semaphores** to manage the printer access, ensuring that no more than 2 users can print at the same time.