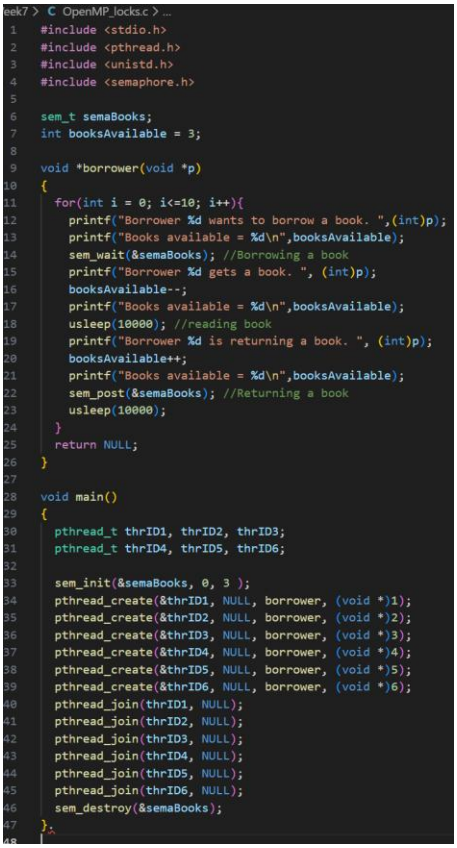
University of Wolverhampton

School of Mathematics and Computer Science

6CS005 High Performance Computing Week 7 Workshop

Tasks – More OpenMP Multithreading

1. The following program uses semaphores to manage the loan of 3 books to a group of 5 borrowers:



```
eek7 > C OpenMP_locks.c > ...
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>

sem_t semaBooks;
int booksAvailable = 3;

void *borrower(void *p)
{
  for(int i = 0; i<=10; i++){
    printf("Borrower %d wants to borrow a book. ",(int)p);
    printf("Books available = %d\n",booksAvailable);
    sem_wait(&semaBooks); //Borrowing a book
    printf("Borrower %d gets a book. ", (int)p);
    booksAvailable--;
    printf("Books available = %d\n",booksAvailable);
    usleep(10000); //reading book
    printf("Borrower %d is returning a book. ", (int)p);
    booksAvailable++;
    printf("Books available = %d\n",booksAvailable);
    sem_post(&semaBooks); //Returning a book
    usleep(10000);
  }
  return NULL;
}

void main()
{
  pthread_t thrID1, thrID2, thrID3;
  pthread_t thrID4, thrID5, thrID6;

  sem_init(&semaBooks, 0, 3 );
  pthread_create(&thrID1, NULL, borrower, (void *)1);
  pthread_create(&thrID2, NULL, borrower, (void *)2);
  pthread_create(&thrID3, NULL, borrower, (void *)3);
  pthread_create(&thrID4, NULL, borrower, (void *)4);
  pthread_create(&thrID5, NULL, borrower, (void *)5);
  pthread_create(&thrID6, NULL, borrower, (void *)6);
  pthread_join(thrID1, NULL);
  pthread_join(thrID2, NULL);
  pthread_join(thrID3, NULL);
  pthread_join(thrID4, NULL);
  pthread_join(thrID5, NULL);
  pthread_join(thrID6, NULL);
  sem_destroy(&semaBooks);
};
```

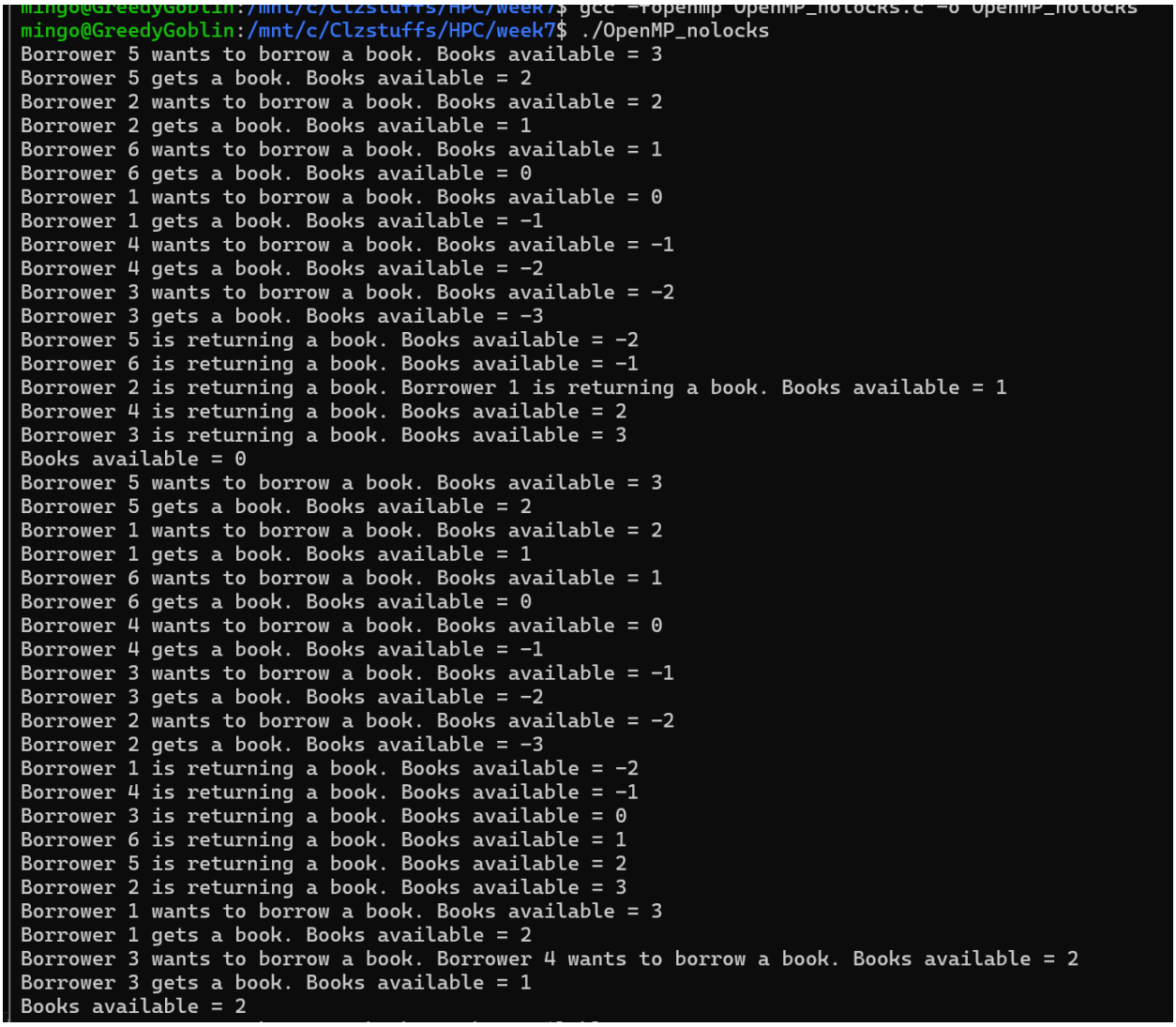
a. Convert it to use OpenMP locks.

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ cat OpenMP_Locks.c
#include <stdio.h>
#include <omp.h>
#include <unistd.h>

omp_lock_t lockBooks;
int booksAvailable = 3;

void borrower(int p)
{
    for(int i = 0; i <= 10; i++) {
        printf("Borrower %d wants to borrow a book. Books available = %d\n", p, booksAvailable);

        omp_set_lock(&lockBooks);
        printf("Borrower %d gets a book. Books available = %d\n", p, booksAvailable);
        booksAvailable--;
        omp_unset_lock(&lockBooks);

        usleep(10000);

        omp_set_lock(&lockBooks);
        printf("Borrower %d is returning a book. Books available = %d\n", p, booksAvailable);
        booksAvailable++;
        omp_unset_lock(&lockBooks);

        usleep(10000);
    }
}

int main()
{
    omp_init_lock(&lockBooks);

    #pragma omp parallel num_threads(5)
    {
        int id = omp_get_thread_num() + 1;
        borrower(id);
    }

    omp_destroy_lock(&lockBooks);
    return 0;
}
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$
```

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ ./OpenMP_Locks
Borrower 1 wants to borrow a book. Books available = 3
Borrower 1 gets a book. Books available = 3
Borrower 2 wants to borrow a book. Books available = 3
Borrower 2 gets a book. Books available = 2
Borrower 5 wants to borrow a book. Books available = 3
Borrower 5 gets a book. Books available = 1
Borrower 3 wants to borrow a book. Books available = 3
Borrower 3 gets a book. Books available = 0
Borrower 4 wants to borrow a book. Books available = 3
Borrower 4 gets a book. Books available = -1
Borrower 1 is returning a book. Books available = -2
Borrower 5 is returning a book. Books available = -1
Borrower 2 is returning a book. Books available = 0
Borrower 4 is returning a book. Books available = 1
Borrower 3 is returning a book. Books available = 2
Borrower 1 wants to borrow a book. Books available = 3
Borrower 1 gets a book. Books available = 3
Borrower 5 wants to borrow a book. Books available = 2
Borrower 5 gets a book. Books available = 2
Borrower 4 wants to borrow a book. Books available = 1
Borrower 4 gets a book. Books available = 1
Borrower 2 wants to borrow a book. Books available = 0
Borrower 2 gets a book. Books available = 0
Borrower 3 wants to borrow a book. Books available = 0
Borrower 3 gets a book. Books available = -1
Borrower 5 is returning a book. Books available = -2
Borrower 4 is returning a book. Books available = -1
Borrower 1 is returning a book. Books available = 0
Borrower 3 is returning a book. Books available = 1
Borrower 2 is returning a book. Books available = 2
Borrower 5 wants to borrow a book. Books available = 3
Borrower 5 gets a book. Books available = 3
Borrower 4 wants to borrow a book. Books available = 3
Borrower 4 gets a book. Books available = 2
Borrower 1 wants to borrow a book. Books available = 1
Borrower 1 gets a book. Books available = 1
Borrower 3 wants to borrow a book. Books available = 1
Borrower 3 gets a book. Books available = 0
Borrower 2 wants to borrow a book. Books available = -1
Borrower 2 gets a book. Books available = -1
Borrower 5 is returning a book. Books available = -2
Borrower 4 is returning a book. Books available = -1
Borrower 1 is returning a book. Books available = 0
Borrower 3 is returning a book. Books available = 1
```

b. Modify the program so that it doesn't use locks.

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ cat OpenMP_nolocks.c
#include <stdio.h>
#include <omp.h>
#include <unistd.h>

int booksAvailable = 3;

void borrower(int p)
{
    for (int i = 0; i <= 10; i++)
    {
        printf("Borrower %d wants to borrow a book. ", p);
        printf("Books available = %d\n", booksAvailable);

        printf("Borrower %d gets a book. ", p);
        booksAvailable--;
        printf("Books available = %d\n", booksAvailable);

        usleep(10000);

        printf("Borrower %d is returning a book. ", p);
        booksAvailable++;
        printf("Books available = %d\n", booksAvailable);

        usleep(10000);
    }
}

int main()
{
    #pragma omp parallel num_threads(6)
    {
        int id = omp_get_thread_num() + 1;
        borrower(id);
    }

    return 0;
}
```



2.The following program encodes 3 lower case letters into a numeric code:

Enter it as "encode.c", build and run it and then enter 3 lower case letters. Note down the code it produces

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ cat encode.c
#include <stdio.h>

long encode(char *s)
{
  long a,b,c,x;
  a = s[0];
  b = s[1];
  c = s[2];
  x = ((((a*69)+c)*137)+b)*39;
  x = x % 54321;
  return x;
}

void main()
{
  char s[100];
  long x;
  printf("Enter 3 lowercase letters: ");
  scanf("%s",s);
  s[3]='\0';
  x=encode(s);
  printf("Code for %s is %ld\n",s,x);
}

mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ gcc -fopenmp encode.c -o encode
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ ./encode
Enter 3 lowercase letters: xyz
Code for xyz is 27459
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$
```

The next program attempts to decode the code back to the original 3 letters.:

Week7 > C decode.c > main()

```c
#include <stdio.h>
#include <stdlib.h>

long encode(char *s)
{
  long a,b,c,x;
  a = s[0];
  b = s[1];
  c = s[2];
  x = ((((a*69)+c)*137)+b)*39;
  x = x % 54321;
  return x;
}

void main()
{
  char s[4];
  long x,y;
  int i,j,k;
  printf("Enter the code: ");
  scanf("%ld",&x);
  s[3]='\0';
  for(i=0;i<26;i++){
    s[0]=i+'a';
    for(j=0;j<26;j++){
      s[1]=j+'a';
      for(k=0;k<26;k++){
        s[2]=k+'a';
        y=encode(s);
        if(x==y){
          printf("The letters for code %ld are %s\n",y,s);
          exit(0);
        }
      }
    }
  }
}

```

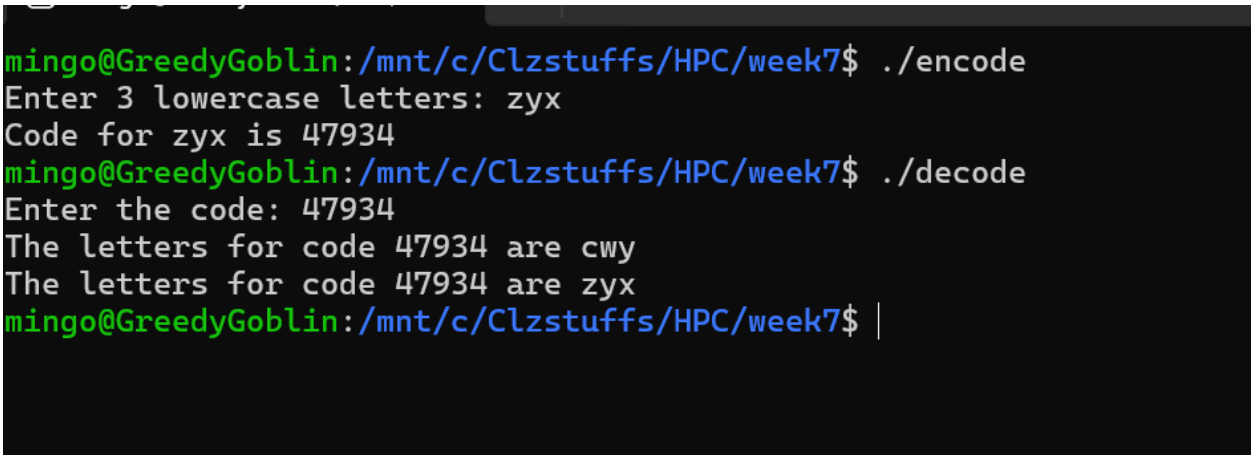
a.Enter it as "decode.c", build and run it and verify that it decodes the letters successfully from the numeric generated by the previous program.

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ cat decode.c
#include <stdio.h>
#include <stdlib.h>

long encode(char *s)
{
  long a,b,c,x;
  a = s[0];
  b = s[1];
  c = s[2];
  x = ((((a*69)+c)*137)+b)*39;
  x = x % 54321;
  return x;
}

void main()
{
  char s[4];
  long x,y;
  int i,j,k;
  printf("Enter the code: ");
  scanf("%ld",&x);
  s[3]='\0';
  for(i=0;i<26;i++){
    s[0]=i+'a';
    for(j=0;j<26;j++){
      s[1]=j+'a';
      for(k=0;k<26;k++){
        s[2]=k+'a';
        y=encode(s);
        if(x==y){
          printf("The letters for code %ld are %s\n",y,s);
          exit(0);
        }
      }
    }
  }
}
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ ./decode
Enter the code: 12345
The letters for code 12345 are obn
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$
```

Modify the program to use the OpenMP for loop "collapse" method to spread the work across 16 threads.

```c
#include <stdlib.h>
#include <omp.h>

long encode(char *s)
{
    long a,b,c,x;
    a = s[0];
    b = s[1];
    c = s[2];
    x = ((((a*69)+c)*137)+b)*39;
    x = x % 54321;
    return x;
}

int main()
{
    char s[4];
    long x, y;

    printf("Enter the code: ");
    scanf("%ld", &x);

    s[3] = '\0';

    #pragma omp parallel for collapse(3) num_threads(16) private(s, y)
    for (int i = 0; i < 26; i++) {
        for (int j = 0; j < 26; j++) {
            for (int k = 0; k < 26; k++) {

                s[0] = i + 'a';
                s[1] = j + 'a';
                s[2] = k + 'a';

                y = encode(s);

                if (x == y) {
                    printf("The letters for code %ld are %s\n", y, s);
                    exit(0);
                }
            }
        }
    }

    return 0;
}
```

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$
```


mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ ./encode
Enter 3 lowercase letters: zyx
Code for zyx is 47934
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$ ./decode
Enter the code: 47934
The letters for code 47934 are cwy
The letters for code 47934 are zyx
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/week7$