Zadanie 2.
Program skompilował się po użyciu komendy :
gcc -o hello hello32.c -pthread -lm oraz zamianie w 28 linijce kodu %ln na %p.
Linijka kodu po zmianie:
`printf("%p: Hello World!\n", threadid);`

Kolejność w jakiej wyświetlane są wyjścia wątków jest za każdym razem inna. Jest to spowodowane tym, że planista jest często wywoływany oraz wybiera proces z puli procesów gotowych i przydziela mu procesor. Ponadto wpływ ma to że planista musi cały czas na bieżąco przydzielać zasoby dla wątku, jednocześnie wykonując obliczenia dla innego wątku.

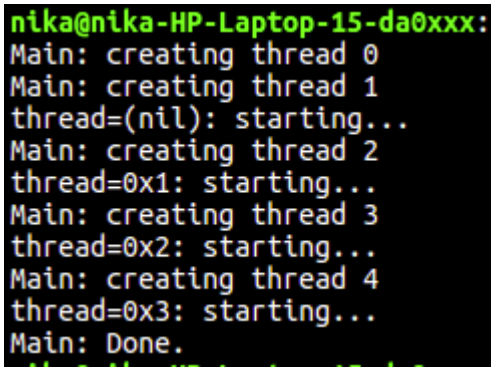
Zadanie 3.
Początkowo program bug3.c wyświetlał napis Hello from thread 8. Po wprowadzonych poprawkach w kodzie wyświetla Hello from thread i daje różne numery wątków.

W funkcji PrintHello(void *threadid) zastąpiono linijkę :
taskid = *(long *)threadid na taskid = (long) treadid
Oprócz tego w funkcji main stworzono tablicę :
long taskids [NUM_THREADS] i odwołano się do niej w pętli for oraz w funkcji pthread_create.

```
void *PrintHello(void *threadid)
{
   long taskid;
   sleep(1);
   taskid = (long)threadid;
   printf("Hello from thread %ld\n", taskid);
   pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
pthread_t threads[NUM_THREADS];
long taskids[NUM_THREADS];
int rc;
long t;

for(t=0;t<NUM_THREADS;t++) {
   taskids[t]=t;
  printf("Creating thread %ld\n", t);
  rc = pthread_create(&threads[t], NULL, PrintHello, (void *) taskids[t]);
  if (rc) {
    printf("ERROR; return code from pthread_create() is %d\n", rc);
    exit(-1);
    }
   }

pthread_exit(NULL);
}
```

Zadanie 4.

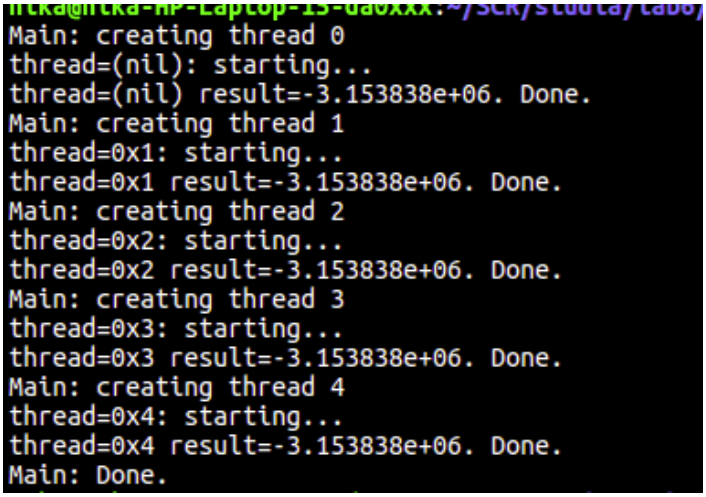
Efekt po uruchomieniu programu:



Program nie działa prawidłowo, ponieważ kończy się szybciej niż wykonają się wszystkie wątki. Aby to naprawić należało w main dodać funkcję int pthread_join(pthread_t thread, void **retval). Jest to funkcja czekająca na zamknięcie wątku. Jako pierwszy argument podajemy wątek na który czekamy a jako drugi wartość zwracaną przez wątek.

```
int main(int argc, char *argv[])
{
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t=0;t<NUM_THREADS;t++){
  printf("Main: creating thread %ld\n", t);
  rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
  pthread_join(threads[t],NULL);
  if (rc){
    printf("ERROR; return code from pthread_create() is %d\n", rc);
    exit(-1);
    }
  }
printf("Main: Done.\n");
}
```

Efekt po wprowadzeniu zmian:



Wszystkie utworzone wątki zdążą się wykonać.

Zadanie 5.

Zmieniono linjkę 28 kodu programu join.c po to aby wątki w zwracały różne wartości. Modyfikacja polegała na tym że w nawiasie pomnożono wartość result o wartośc tid.

```
#include <math.h>
#define NUM_THREADS  4

void *BusyWork(void *t)
{
   int i;
   long tid;
   double result=0.0;
   tid = (long)t;
   printf("Thread %ld starting...\n",tid);
   for (i=0; i<1000000; i++)
   {
      result = result + sin(i) * tan(i);
   }
   printf("Thread %ld done. Result = %e\n",tid, result*tid);

   pthread_exit((void*) t);
}
```

Działanie programu po modyfikacji:

```
nika@nika-HP-Laptop-15-da0xxx:~/SCR/studia/lab6/kod1$ ./
Main: creating thread 0
Main: creating thread 1
Thread 0 starting...
Main: creating thread 2
Thread 1 starting...
Main: creating thread 3
Thread 2 starting...
Thread 3 starting...
Thread 1 done. Result = -3.153838e+06
Thread 3 done. Result = -9.461513e+06
Thread 2 done. Result = -6.307675e+06
Thread 0 done. Result = -0.000000e+00
Main: completed join with thread 0 having a status of 0
Main: completed join with thread 1 having a status of 1
Main: completed join with thread 2 having a status of 2
Main: completed join with thread 3 having a status of 3
Main: program completed. Exiting.
```

Różnica pomiędzy programami join.c a detached.c oraz ich sposobem działania wynika z tego, że oba te programy używają innego rodzaju wątków. Program join.c korzysta z wątków joinable. Są to wątki, które w momencie zakończenia pracy nie zwalniają zasobów, aż do chwili wywołania funkcji pthread_join, która jest odpowiedzialna za oczekiwanie na zakończenie wątku. Utworzenie wątku typu joinable i nie wywołanie funkcji pthread_join spowoduje wyciek pamięci. Program detached.c używa wątków typu detached. Wątki te zwalniają od razu wszystkie zasoby w momencie zakończenia działania.