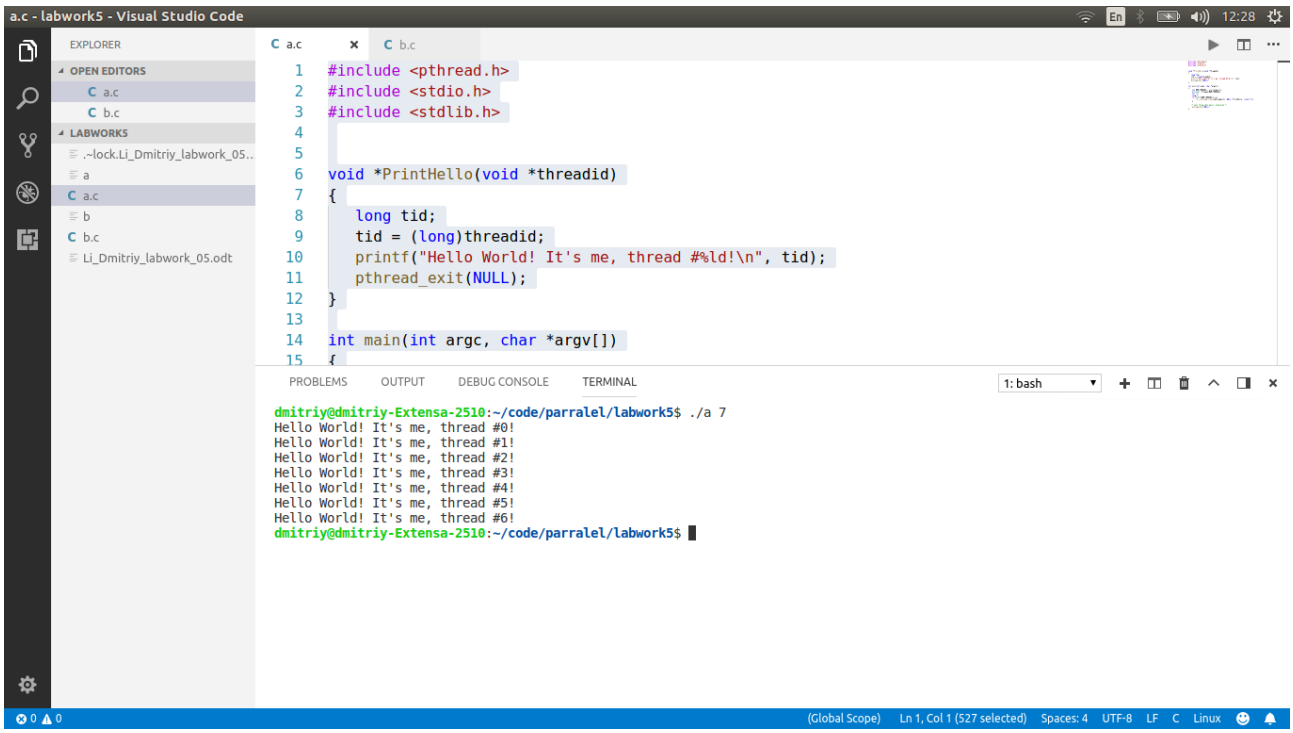
1. Write a Hello World program using Pthreads.

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


void *PrintHello(void *threadid)
{
long tid;
tid = (long)threadid;
printf("Hello World! It's me, thread #%ld!\n", tid);
pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
int NUM_THREADS = atoi(argv[1]);
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t=0;t<NUM_THREADS;t++){
rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
}

/* Last thing that main() should do */
pthread_exit(NULL);
}
```



2.Implement a matrix-vector multiplication using Pthreads.

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

#define M 3
#define K 2
#define N 3

int A [M][K];
int B [K][N];
int C [M][N];

struct v {
int i;
int j;
};
```

```
void *runner(void *param);

int main(int argc, char *argv[])
{

for(int i = 0; i < M;i++)
{
printf("Enter %d raw\n", i+1);
for(int j = 0; j < K;j++)
{
scanf("%d",&A[i][j]);
// B[j] [i] =rand()%20;
}
}

for(int i = 0; i < K;i++)
{
printf("Enter %d raw\n", i+1);
for(int j = 0; j < N;j++)
{
scanf("%d",&B[i][j]);
// B[j] [i] =rand()%20;
}
}

int i,j, count = 0;
for(i = 0; i < M; i++) {
for(j = 0; j < N; j++) {

struct v *data = (struct v *) malloc(sizeof(struct v));
data->i = i;
data->j = j;

pthread_t tid;
pthread_attr_t attr;

pthread_attr_init(&attr);

pthread_create(&tid,&attr,runner,data);

pthread_join(tid, NULL);
count++;
}
}
```

```
for(i = 0; i < M; i++) {
for(j = 0; j < N; j++) {
printf("%d ", C[i][j]);
}
printf("\n");
}
}
```

```
void *runner(void *param) {
struct v *data = param;
int n, sum = 0;
```

```
for(n = 0; n< K; n++){
sum += A[data->i][n] * B[n][data->j];
}
C[data->i][data->j] = sum;
pthread_exit(0);
}
```