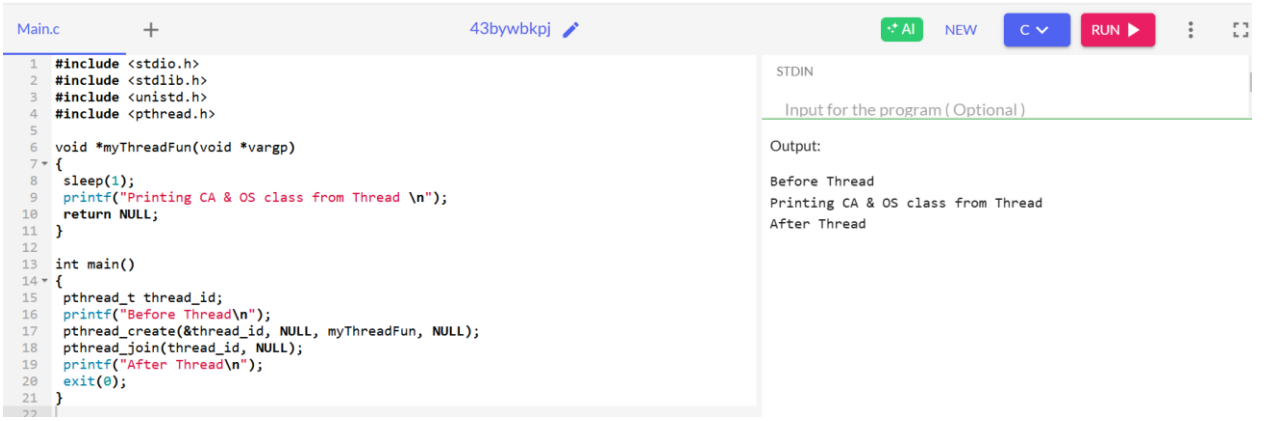
PRACTICE 5

1.



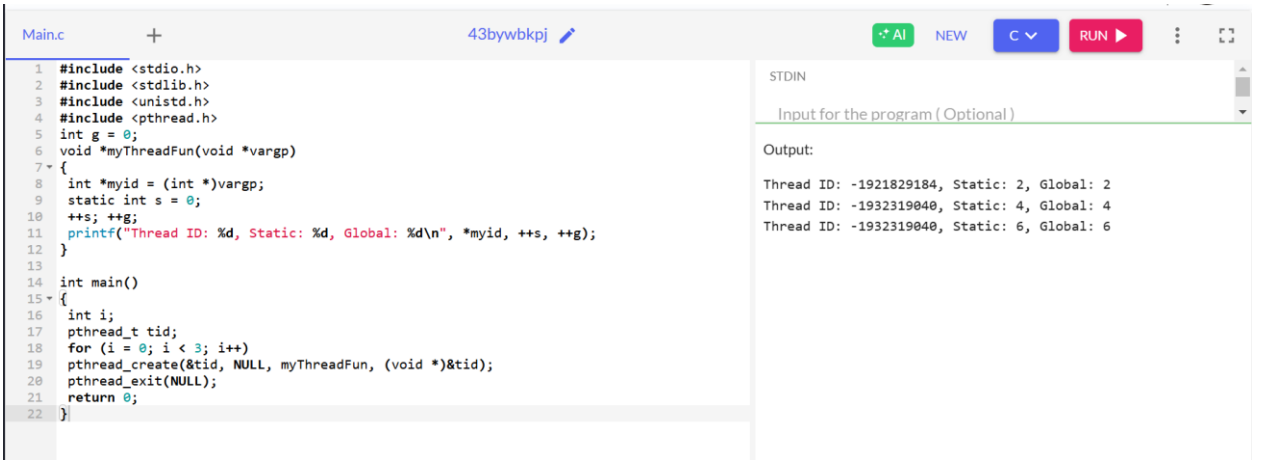
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

void *myThreadFun(void *vargp)
{
 sleep(1);
 printf("Printing CA & OS class from Thread \n");
 return NULL;
}

int main()
{
 pthread_t thread_id;
 printf("Before Thread\n");
 pthread_create(&thread_id, NULL, myThreadFun, NULL);
 pthread_join(thread_id, NULL);
 printf("After Thread\n");
 exit(0);
}
```

```
Before Thread
Printing CA & OS class from Thread
After Thread
```

2.

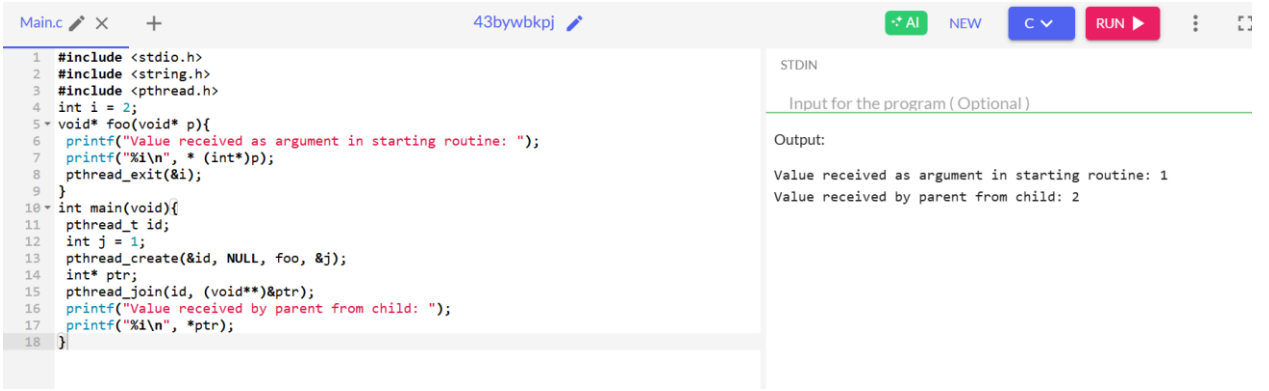


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
int g = 0;
void *myThreadFun(void *vargp)
{
 int *myid = (int *)vargp;
 static int s = 0;
 ++s; ++g;
 printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
 int i;
 pthread_t tid;
 for (i = 0; i < 3; i++)
 pthread_create(&tid, NULL, myThreadFun, (void *)&tid);
 pthread_exit(NULL);
 return 0;
}
```

```
Thread ID: -1921829184, Static: 2, Global: 2
Thread ID: -1932319040, Static: 4, Global: 4
Thread ID: -1932319040, Static: 6, Global: 6
```

3.



```c
#include <stdio.h>
#include <string.h>
#include <pthread.h>
int i = 2;
void* foo(void* p){
 printf("Value received as argument in starting routine: ");
 printf("%i\n", * (int*)p);
 pthread_exit(&i);
}
int main(void){
 pthread_t id;
 int j = 1;
 pthread_create(&id, NULL, foo, &j);
 int* ptr;
 pthread_join(id, (void**)&ptr);
 printf("Value received by parent from child: ");
 printf("%i\n", *ptr);
}
```

```
Value received as argument in starting routine: 1
Value received by parent from child: 2
```

4.

```c
#include <stdio.h>
#include <string.h>
#include <pthread.h>
int i = 2;
void* foo(void* p){
 printf("Value received as argument in starting routine:");
 printf("%i\n", * (int*)p);
 pthread_exit(&i); }
int main(void){
 pthread_t id;
 int j = 1;
 pthread_create(&id, NULL, foo, &j);
 int* ptr;
 pthread_join(id, (void**)&ptr);
 printf("Value received by parent from child: ");
 printf("%i\n", *ptr);
}
```

```
Value received as argument in starting routine:1
Value received by parent from child: 2
```

There is a global variable i = 2. The child thread will return this value.

The thread function takes an argument, prints its value, and then exits. It returns a pointer to the global variable i.

In the main function, a thread is created and gets j = 1 as an argument.
The parent process waits for the child thread to finish. It receives a pointer to i.
The program prints two values:
The child thread prints 1 (the value of j).
The parent prints 2 (the value of i from the child).