CS266 PRACTICE LAB 4

NAME:

ARCHIT AGRAWAL

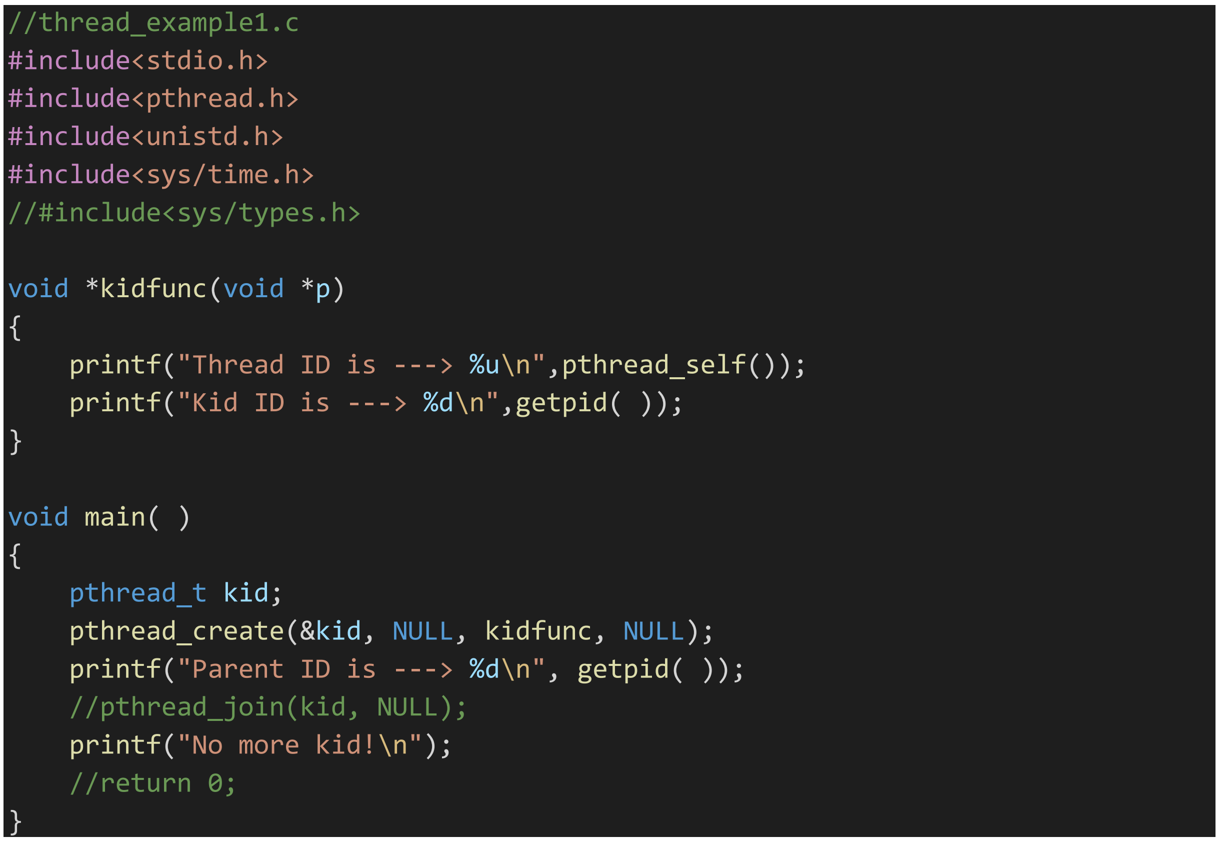
ROLL NO. :

202051213

SECTION:

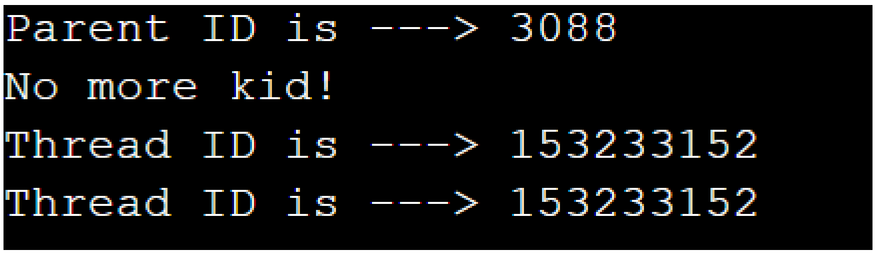
2

Code 1

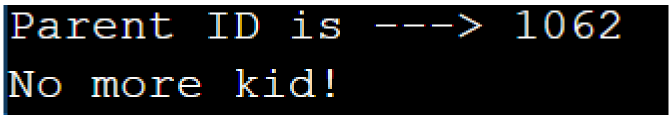


```c
//thread_example1.c
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
#include<sys/time.h>
//#include<sys/types.h>

void *kidfunc(void *p)
{
    printf("Thread ID is ---> %u\n",pthread_self());
    printf("Kid ID is ---> %d\n",getpid( ));
}

void main( )
{
    pthread_t kid;
    pthread_create(&kid, NULL, kidfunc, NULL);
    printf("Parent ID is ---> %d\n", getpid( ));
    //pthread_join(kid, NULL);
    printf("No more kid!\n");
    //return 0;
}
```

Output

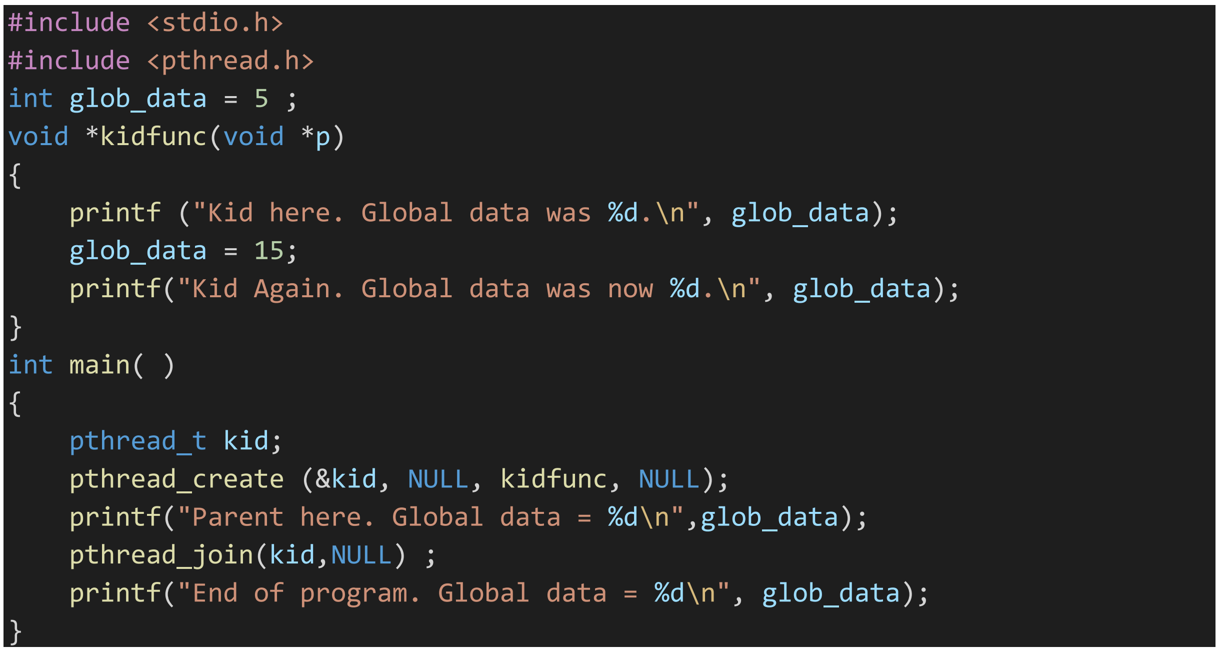


```
Parent ID is ---> 3088
No more kid!
Thread ID is ---> 153233152
Thread ID is ---> 153233152
```



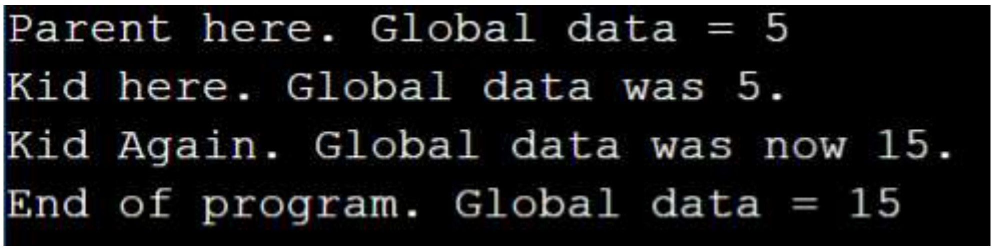
```
Parent ID is ---> 1062
No more kid!
```

Code 2



```c
#include <stdio.h>
#include <pthread.h>
int glob_data = 5 ;
void *kidfunc(void *p)
{
    printf ("Kid here. Global data was %d.\n", glob_data);
    glob_data = 15;
    printf("Kid Again. Global data was now %d.\n", glob_data);
}
int main( )
{
    pthread_t kid;
    pthread_create (&kid, NULL, kidfunc, NULL);
    printf("Parent here. Global data = %d\n",glob_data);
    pthread_join(kid,NULL) ;
    printf("End of program. Global data = %d\n", glob_data);
}
```

Output

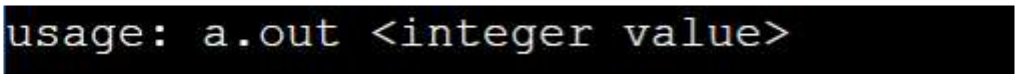


```
Parent here. Global data = 5
Kid here. Global data was 5.
Kid Again. Global data was now 15.
End of program. Global data = 15
```

Code 3

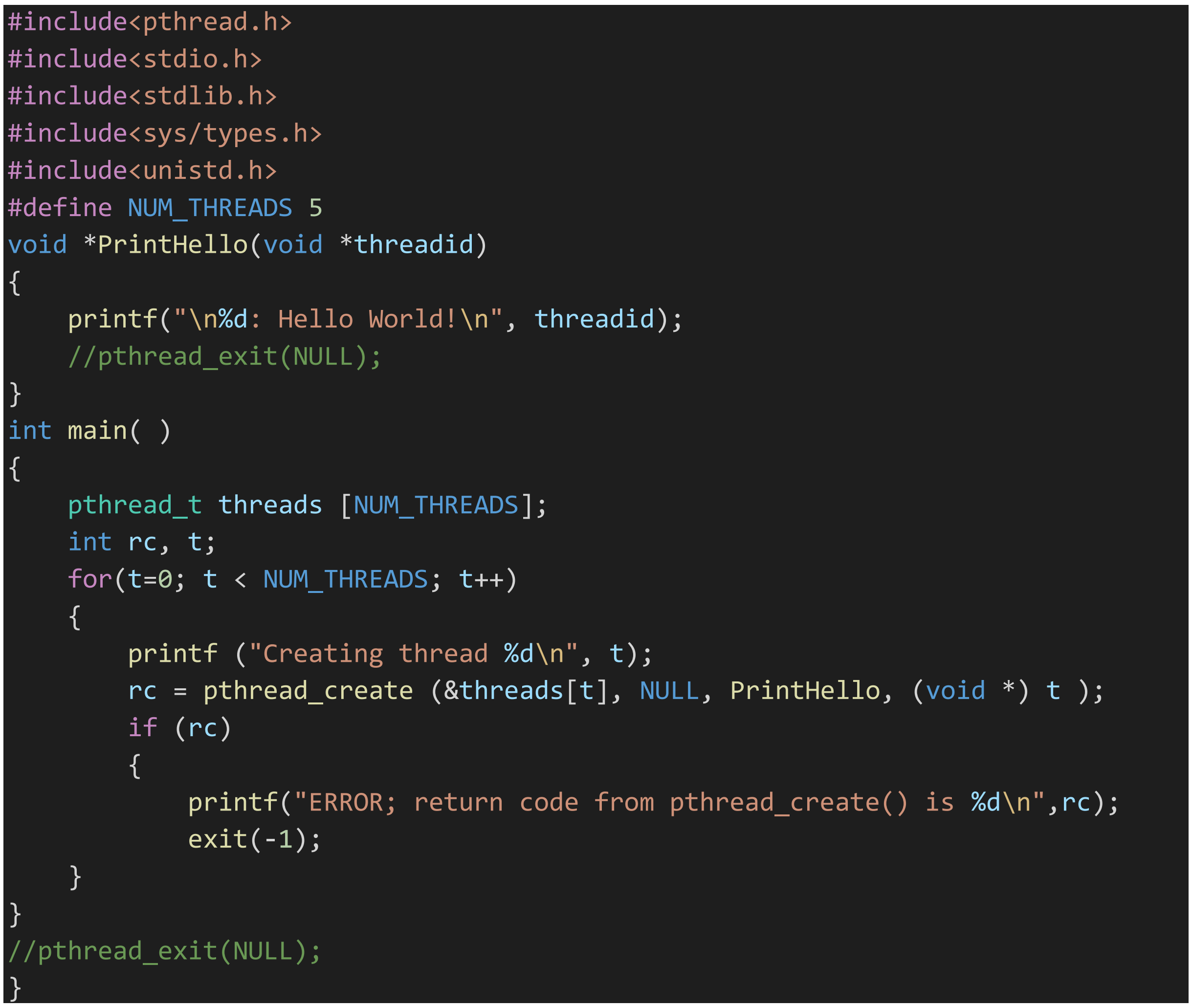
```c
#include<pthread.h>
#include<stdio.h>
//#include<sys/types.h>
//#include<unistd.h>
#include<stdlib.h>
int sum; /*This data is shared by the thread(s) */
void *runner(void *param); /* the thread */
void main(int argc, char *argv[]){
    pthread_t tid; /* the thread identifier */
    pthread_attr_t attr; /* set of thread attributes */
    if(argc != 2)
    {
        fprintf(stderr,"usage: a.out <integer value>\n");
        exit(0);
    }
    if(atoi(argv[1]) < 0)
    {
        fprintf(stderr, "%d must be >= 0 \n", atoi(argv[1]));
        exit(0);
    }
/* get the default attributes */
pthread_attr_init(&attr);

/*create the thread */
pthread_create(&tid,&attr,runner,argv[1]);

/* Now wait for the thread to exit */
pthread_join(tid,NULL);
printf("sum = %d\n",sum);
}

/*The thread will begin control in this function */
void *runner(void *param)
{
    int upper = atoi(param);
    int i;
    sum=0;
    if(upper > 0)
    {
        for(i=1; i <= upper;i++)
        sum += i;
    }
    pthread_exit(0);
}
```

Output



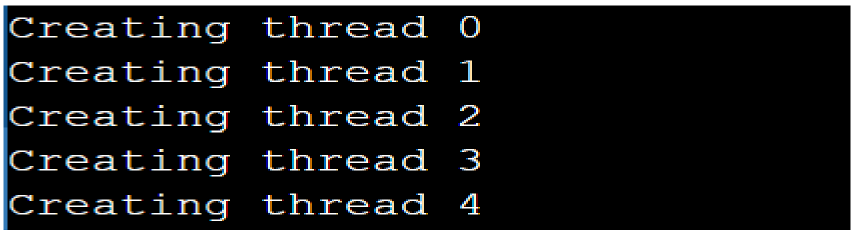
```
usage: a.out <integer value>
```

Code 4



```c
#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
#define NUM_THREADS 5
void *PrintHello(void *threadid)
{
    printf("\n%d: Hello World!\n", threadid);
    //pthread_exit(NULL);
}
int main( )
{
    pthread_t threads [NUM_THREADS];
    int rc, t;
    for(t=0; t < NUM_THREADS; t++)
    {
        printf ("Creating thread %d\n", t);
        rc = pthread_create (&threads[t], NULL, PrintHello, (void *) t );
        if (rc)
        {
            printf("ERROR; return code from pthread_create() is %d\n",rc);
            exit(-1);
    }
}
//pthread_exit(NULL);
}
```

Output

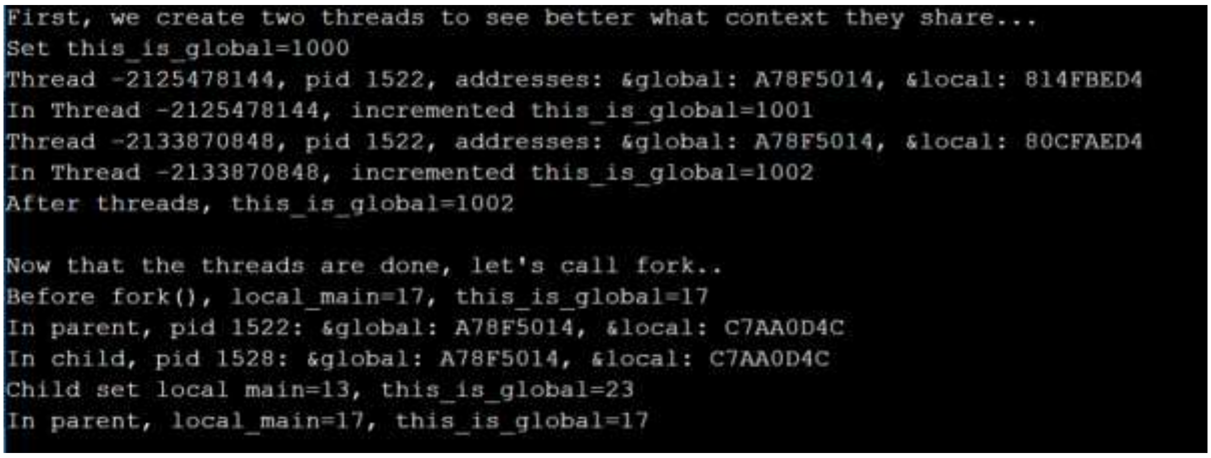


```
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3
Creating thread 4
```

Code 5

```c
//thread_example5.c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include<stdlib.h>
int this_is_global;
void thread_func( void *ptr );
int main( )
{
    int local_main; int pid, status;
    pthread_t thread1, thread2;
    printf("First, we create two threads to see better what context they
share...\n");
    this_is_global=1000;
    printf("Set this_is_global=%d\n",this_is_global);
    pthread_create( &thread1, NULL, (void*)&thread_func, (void*) NULL);
    pthread_create(&thread2, NULL, (void*)&thread_func, (void*) NULL);
    pthread_join(thread1, NULL); pthread_join(thread2, NULL);
    printf("After threads, this_is_global=%d\n",this_is_global);
    printf("\n");
    printf("Now that the threads are done, let's call fork..\n");
    local_main=17; this_is_global=17;
    printf("Before fork(), local_main=%d, this_is_global=%d\n",local_main,
    this_is_global);
    pid=fork();
    if (pid == 0)
    {
        /* this is the child */
        printf("In child, pid %d: &global: %X, &local: %X\n", getpid(),
&this_is_global, &local_main);
        local_main=13;
        this_is_global=23;
        printf("Child set local main=%d, this_is_global=%d\n",local_main,
this_is_global);
        exit(0);
    }
    else
    {
        /* this is parent */
        printf("In parent, pid %d: &global: %X, &local: %X\n", getpid(),
&this_is_global, &local_main);
        wait(&status);
        printf("In parent, local_main=%d, this_is_global=%d\n",local_main,
this_is_global);
    }
    exit(0);
}

void thread_func(void *dummy)
{
    int local_thread;
    printf("Thread %d, pid %d, addresses: &global: %X, &local: %X\n",
    pthread_self(),getpid(),&this_is_global, &local_thread);
    this_is_global++;
    printf("In Thread %d, incremented this_is_global=%d\n",
pthread_self(),this_is_global);
    pthread_exit(0);
}
```

Output



```
First, we create two threads to see better what context they share...
Set this_is_global=1000
Thread -2125478144, pid 1522, addresses: &global: A78F5014, &local: 814FBED4
In Thread -2125478144, incremented this_is_global=1001
Thread -2133870848, pid 1522, addresses: &global: A78F5014, &local: 80CFAED4
In Thread -2133870848, incremented this_is_global=1002
After threads, this_is_global=1002

Now that the threads are done, let's call fork..
Before fork(), local_main=17, this_is_global=17
In parent, pid 1522: &global: A78F5014, &local: C7AA0D4C
In child, pid 1528: &global: A78F5014, &local: C7AA0D4C
Child set local main=13, this_is_global=23
In parent, local_main=17, this_is_global=17
```