LAB-5

-Saloni Patel

Basics:

Pthread_create:

#include int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);

The pthread_create() function starts a new thread in the calling process. The new thread starts execution by invoking start_routine(); arg is passed as the sole argument of start_routine().

The attr argument points to a pthread_attr_t structure whose contents are used at thread creation time to determine attributes for the new thread; this structure is initialized using pthread_attr_init and related functions. If attr is NULL, then the thread is created with default attributes.

Before returning, a successful call to pthread_create() stores the ID of the new thread in the buffer pointed to by thread; this identifier is used to refer to the thread in subsequent calls to other pthreads functions.

On success, pthread_create() returns 0; on error, it returns an error number, and the contents of thread are undefined.

Pthread_join:

#include int pthread_join(pthread_t thread, void **retval);

Compile and link with -pthread.

The pthread_join() function waits for the thread specified by thread to terminate. If that thread has already terminated, then pthread_join() returns immediately. The thread specified by thread must be joinable.

On success, pthread_join() returns 0; on error, it returns an error number.

Pthread_mutex_lock:

#include int pthread_mutex_lock(pthread_mutex_t *mutex);

The mutex object referenced by mutex shall be locked by calling pthread_mutex_lock(). If the mutex is already locked, the calling thread shall block until the mutex becomes available.

This operation shall return with the mutex object referenced by mutex in the locked state with the calling thread as its owner.

If successful, the pthread_mutex_lock() and pthread_mutex_unlock() functions shall return zero; otherwise, an error number shall be returned to indicate the error.

Pthread_mutex_unlock:

#include int pthread_mutex_unlock(pthread_mutex_t *mutex);

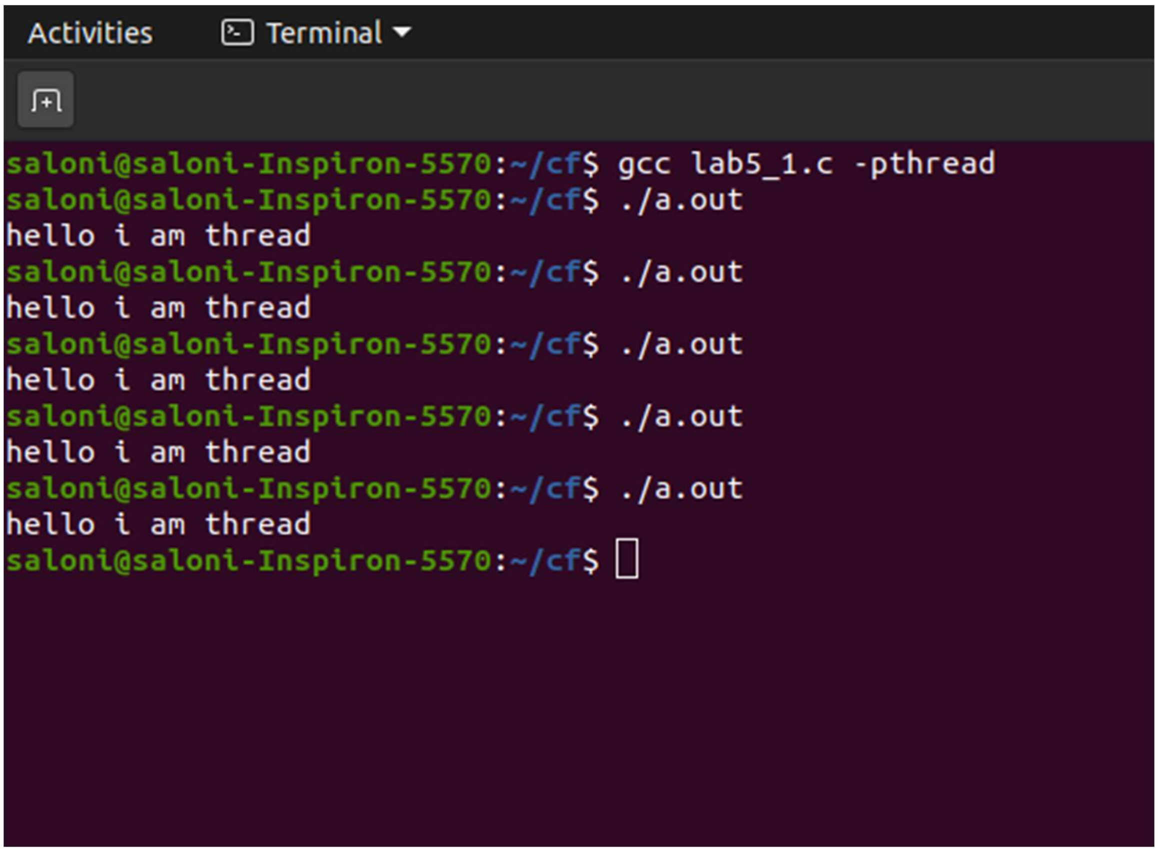
The pthread_mutex_unlock() function shall release the mutex object referenced by mutex.

If successful, the pthread_mutex_lock() and pthread_mutex_unlock() functions shall return zero; otherwise, an error number shall be returned to indicate the error.

1. **Write a program to create a thread using pthread_create.**

```
#include<pthread.h>
#include<stdio.h>
void *f();
void main()
{
        pthread_t t;
        pthread_create(&t,NULL,f,NULL);
        pthread_join(t,NULL);
}
void *f()
{
        printf("hello i am thread\n");
}
```

OUTPUT:



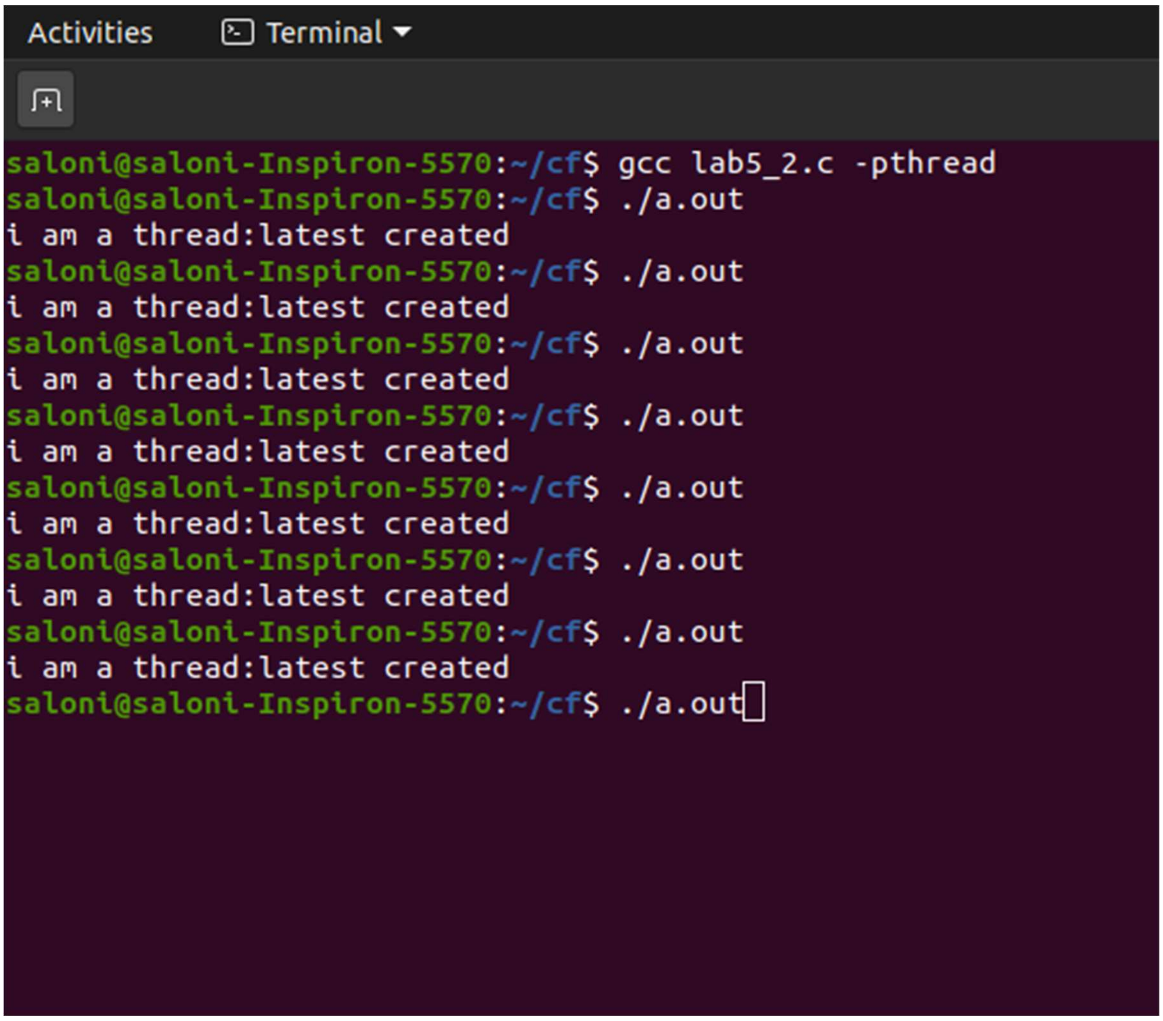
```
saloni@saloni-Inspiron-5570:~/cf$ gcc lab5_1.c -pthread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hello i am thread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hello i am thread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hello i am thread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hello i am thread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
hello i am thread
saloni@saloni-Inspiron-5570:~/cf$ 
```

2. **Write a program to pass a character string to the threaded function.**

```
#include<stdio.h>
#include<pthread.h>
void *f(void *);
void main()
{
        pthread_t t;
```

```
			char *str="latest created";
			pthread_create(&t,NULL,f,(void *)str);
			pthread_join(t,NULL);
}
void *f(void *str)
{
			printf("i am a thread:%s\n",(char *)str);
}
```

OUTPUT:



```
saloni@saloni-Inspiron-5570:~/cf$ gcc lab5_2.c -pthread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
i am a thread:latest created
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
```

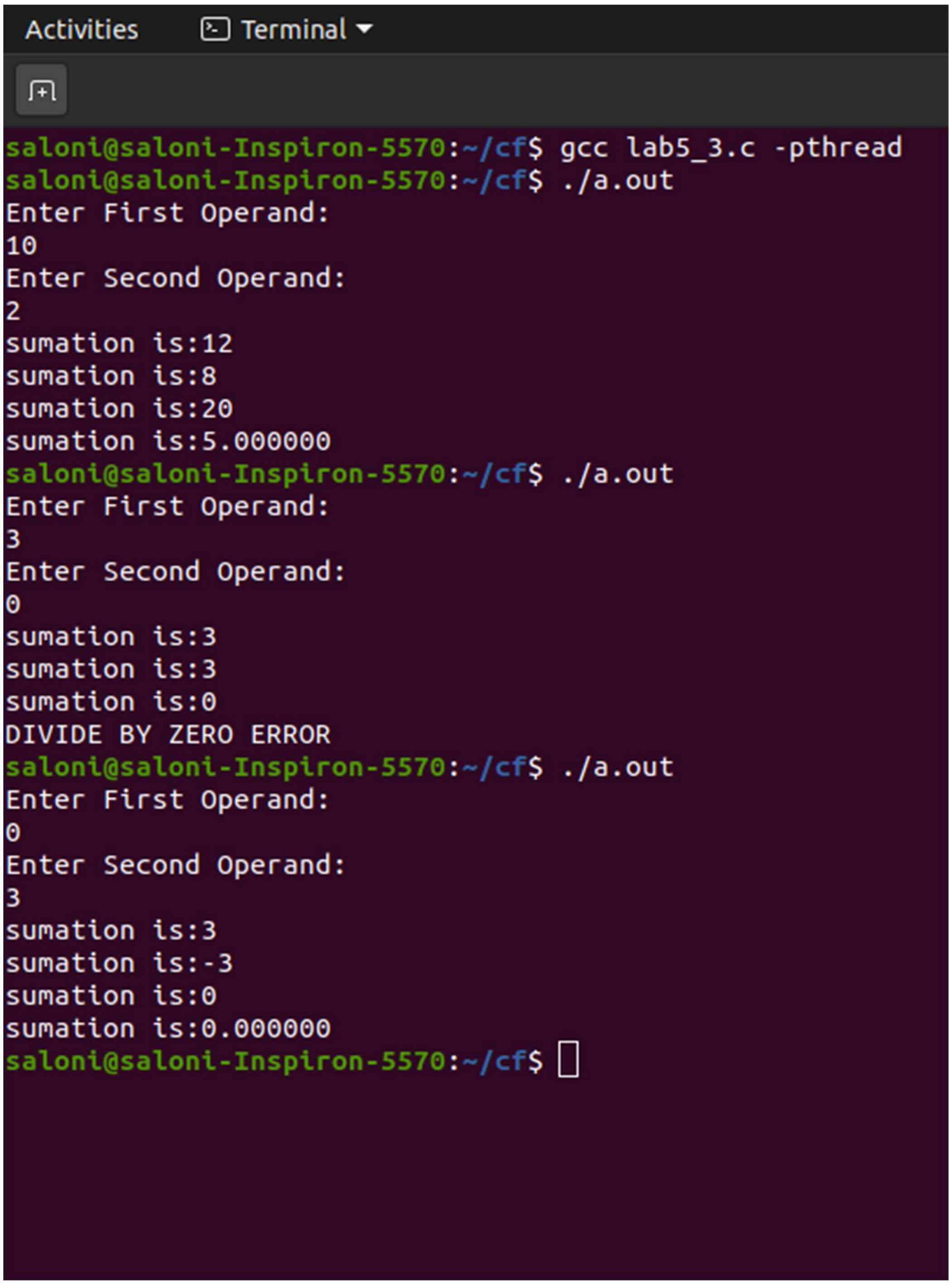
3. **Write a program to implement simple calculator using threads.**

```
#include<stdio.h>
#include<pthread.h>
void *fsum(void *);
void *fsub(void *);
void *fmul(void *);
void *fdiv(void *);

void main()
{
			int arr[2];
```

```
                    printf("Enter First Operand:\n");
                    scanf("%d",&arr[0]);
                    printf("Enter Second Operand:\n");
        scanf("%d",&arr[1]);

                    pthread_t sum;
                    pthread_t sub;
                    pthread_t mul;
                    pthread_t div;

                    pthread_create(&sum,NULL,fsum,(void *)arr);
                    pthread_join(sum,NULL);

                    pthread_create(&sub,NULL,fsub,(void *)arr);
        pthread_join(sub,NULL);

                    pthread_create(&mul,NULL,fmul,(void *)arr);
        pthread_join(mul,NULL);

                    pthread_create(&div,NULL,fdiv,(void *)arr);
        pthread_join(div,NULL);

}
void *fsum(void *temp)
{
                    int *arr=(int *)temp;
                    int ans;
                    ans=arr[0]+arr[1];
                    printf("sumation is:%d\n",ans);
}
void *fsub(void *temp)
{
        int *arr=(int *)temp;
        int ans;
        ans=arr[0]-arr[1];
        printf("sumation is:%d\n",ans);
}
void *fmul(void *temp)
{
        int *arr=(int *)temp;
        int ans;
        ans=arr[0]*arr[1];
        printf("sumation is:%d\n",ans);
}
void *fdiv(void *temp)
{
        int *arr=(int *)temp;
        float ans;
```

```
            if(arr[1]!=0)
            {
            ans=arr[0]/arr[1];
                printf("sumation is:%f\n",ans);
            }
            else
            printf("DIVIDE BY ZERO ERROR\n");
}
```

OUTPUT:



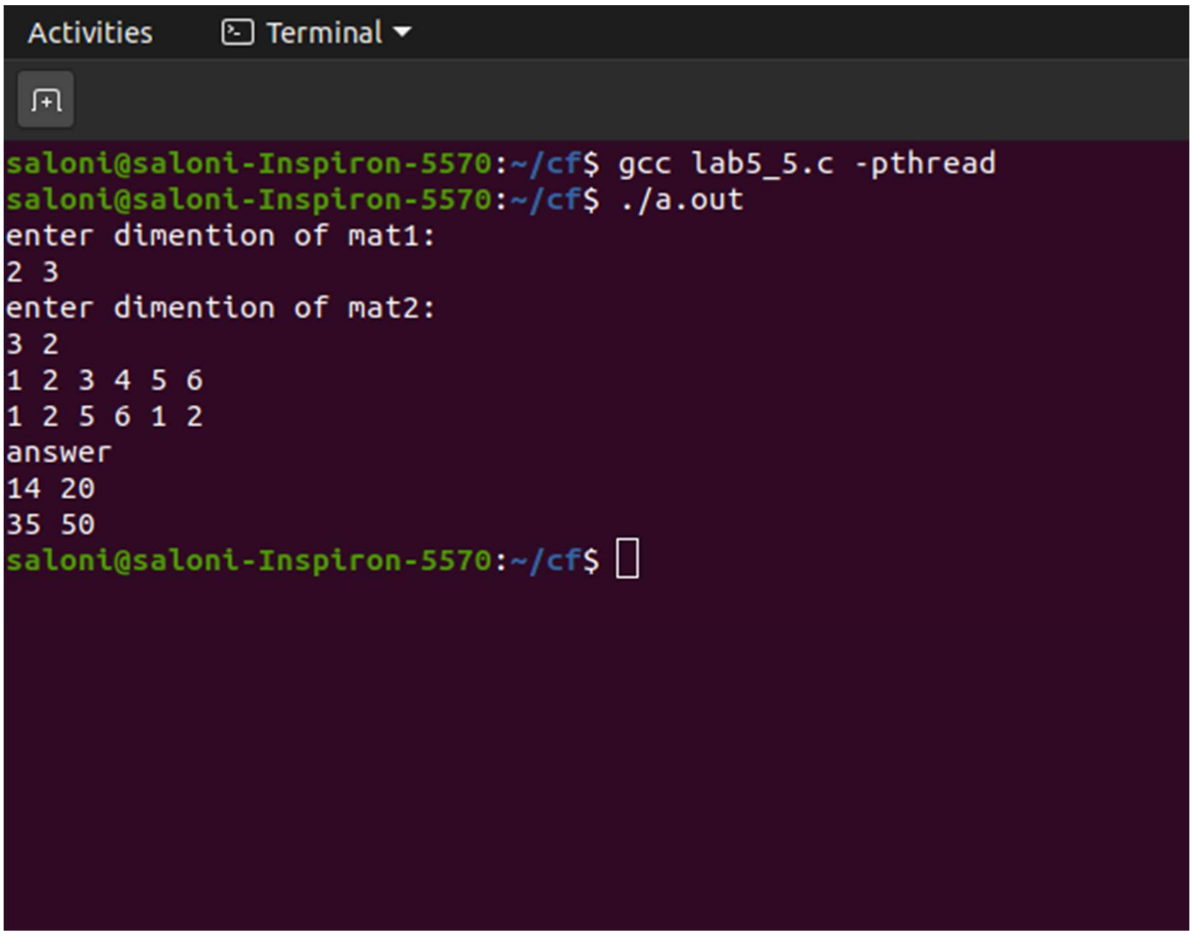
```
saloni@saloni-Inspiron-5570:~/cf$ gcc lab5_3.c -pthread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
Enter First Operand:
10
Enter Second Operand:
2
sumation is:12
sumation is:8
sumation is:20
sumation is:5.000000
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
Enter First Operand:
3
Enter Second Operand:
0
sumation is:3
sumation is:3
sumation is:0
DIVIDE BY ZERO ERROR
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
Enter First Operand:
0
Enter Second Operand:
3
sumation is:3
sumation is:-3
sumation is:0
sumation is:0.000000
saloni@saloni-Inspiron-5570:~/cf$
```

4. **Write a program to multiply two matrices**

```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
int **mat1,**mat2,**res,col,c=0;
void *mul(void *m);
pthread_mutex_t mutex=PTHREAD_MUTEX_INITIALIZER;
typedef struct{
    int rw;
    int cl;
}dim;

void main()
{
                int m,n,x,y,i,j;
                printf("enter dimention of mat1:\n");
                scanf("%d %d",&m,&n);
                printf("enter dimention of mat2:\n");
    scanf("%d %d",&x,&y);
                dim *ptr;
                pthread_t *t;
                mat1=(int **)malloc(sizeof(int)*m);
    for(i=0;i<m;i++){
        mat1[i]=(int *)malloc(sizeof(int)*n);
    }

                for(i=0;i<m;i++)
                {
                     for(j=0;j<n;j++)
                     {
                            scanf("%d",&mat1[i][j]);
                     }
                }
                mat2=(int **)malloc(sizeof(int)*x);
    for(i=0;i<x;i++){
        mat2[i]=(int *)malloc(sizeof(int)*y);
    }

                for(i=0;i<x;i++)
    {
        for(j=0;j<y;j++)
        {
                            scanf("%d",&mat2[i][j]);
                     }
                }
                if(n==x)
                {
                     ptr=(dim *)malloc(sizeof(dim)*m*n);
        t=(pthread_t *)malloc(sizeof(pthread_t)*m*n);
```

```
            col=x;
                              res=(int **)malloc(sizeof(int)*m);

                             for(i=0;i<m;i++){
                   res[i]=(int *)malloc(sizeof(int)*y);
            }

            for(i=0;i<m;i++){
                   for(j=0;j<y;j++)
                                            {
                          ptr[c].rw=i;
                          ptr[c].cl=j;
                          pthread_create(&t[c],NULL,mul,(void *)&ptr[c]);
                          c++;
                   }
            }
            for(i=0;i<c;i++)
            {
                   pthread_join(t[c],NULL);
            }
                             printf("answer\n");
            for(i=0;i<m;i++){
                   for(j=0;j<y;j++){
                          printf("%d ",res[i][j]);
                   }
                   printf("\n");
            }
            free(res);
            free(ptr);
            free(t);
                        }
                        else{
                               printf("not possible\n");

                        }
}
void *mul(void *m){
                        dim *mat=(dim *)m;
       int sum=0,i;
       pthread_mutex_lock(&mutex);
       for(i=0;i<col;i++){
              sum+=mat1[mat->rw][i]*mat2[i][mat->cl];
       }
       res[mat->rw][mat->cl]=sum;
       pthread_mutex_unlock(&mutex);


}
```

OUTPUT:



```
saloni@saloni-Inspiron-5570:~/cf$ gcc lab5_5.c -pthread
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
enter dimention of mat1:
2 3
enter dimention of mat2:
3 2
1 2 3 4 5 6
1 2 5 6 1 2
answer
14 20
35 50
saloni@saloni-Inspiron-5570:~/cf$
```