Họ và tên : Trần Đình Khang

MSSV: 18520072

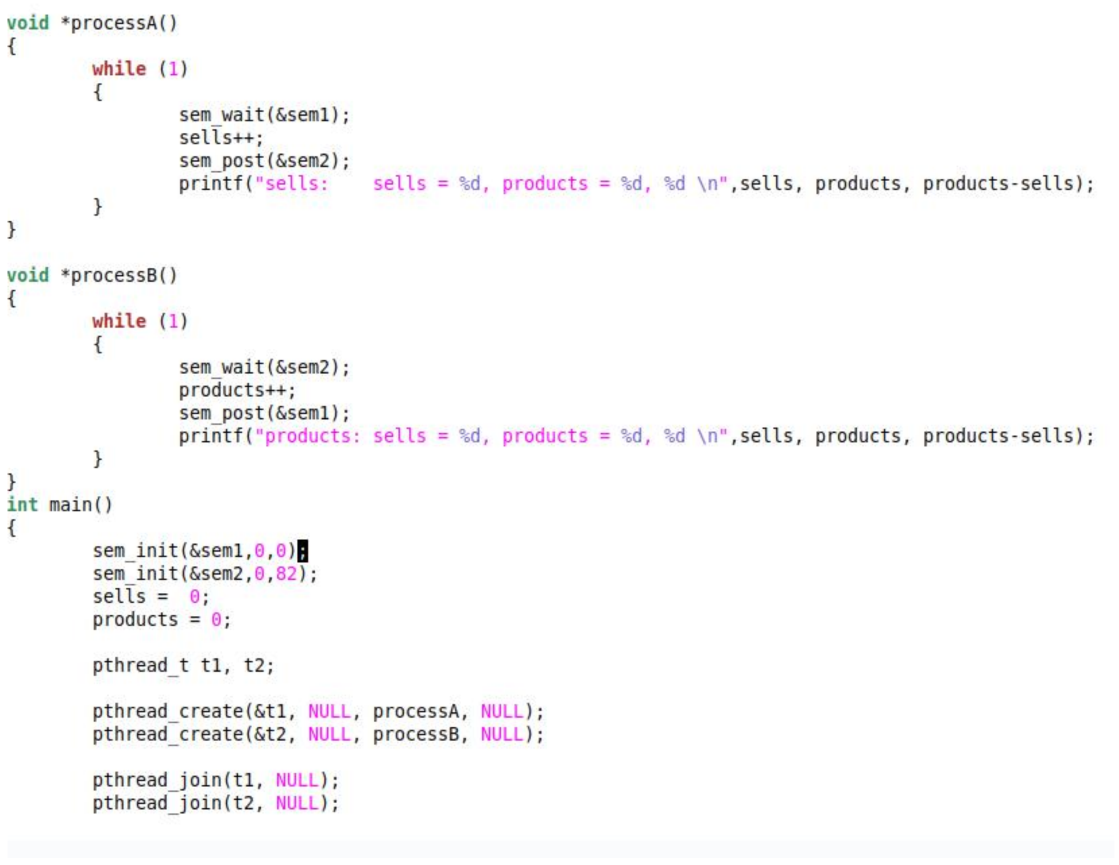
Mã Lớp: IT007.K21.KHTN

Bài thực hành Lab04

5.4 Hướng dẫn thực hành

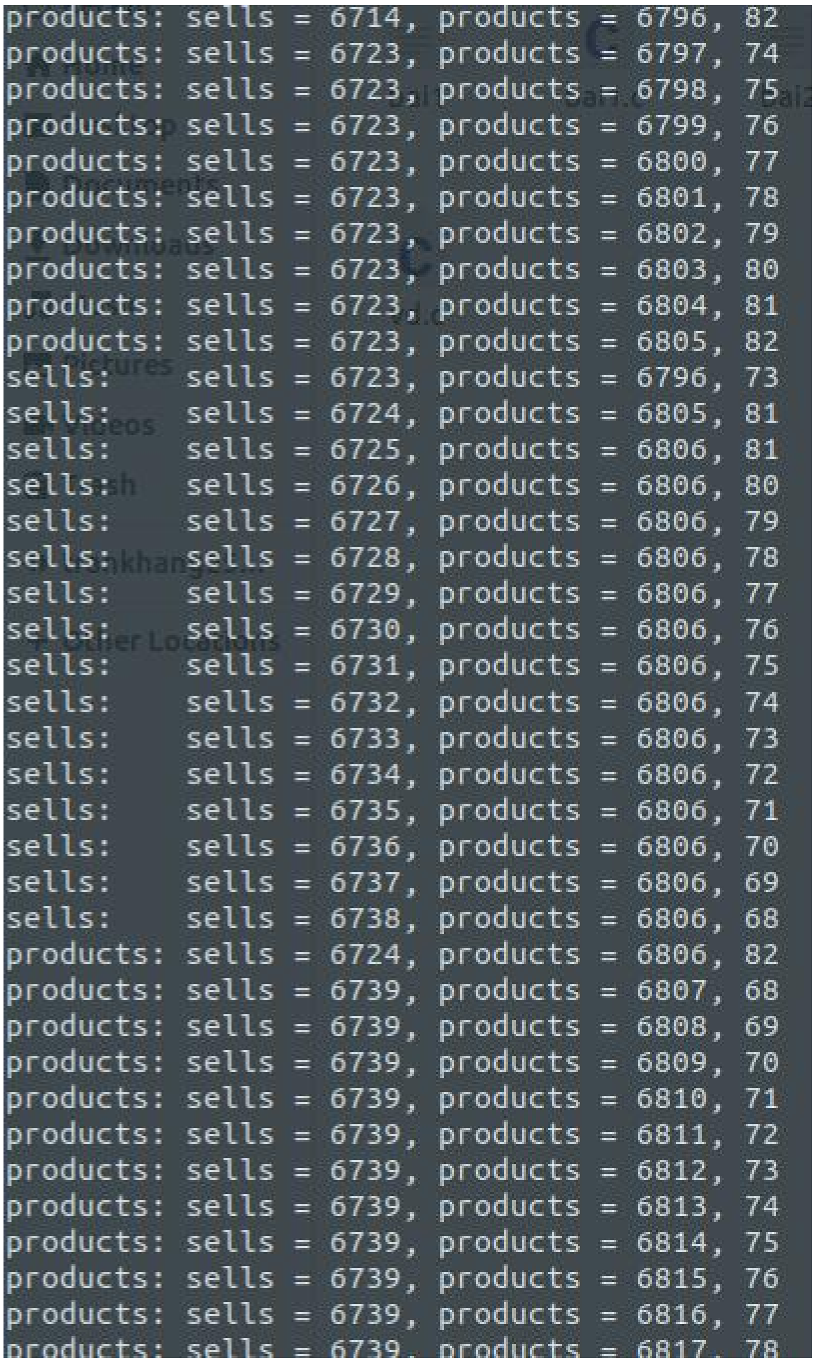
Câu 1:

Code:



```
void *processA()
{
	while (1)
	{
		sem_wait(&sem1);
		sells++;
		sem_post(&sem2);
		printf("sells:    sells = %d, products = %d, %d \n",sells, products, products-sells);
	}
}

void *processB()
{
	while (1)
	{
		sem_wait(&sem2);
		products++;
		sem_post(&sem1);
		printf("products: sells = %d, products = %d, %d \n",sells, products, products-sells);
	}
}
int main()
{
	sem_init(&sem1,0,0);
	sem_init(&sem2,0,82);
	sells =  0;
	products = 0;

	pthread_t t1, t2;

	pthread_create(&t1, NULL, processA, NULL);
	pthread_create(&t2, NULL, processB, NULL);

	pthread_join(t1, NULL);
	pthread_join(t2, NULL);
```

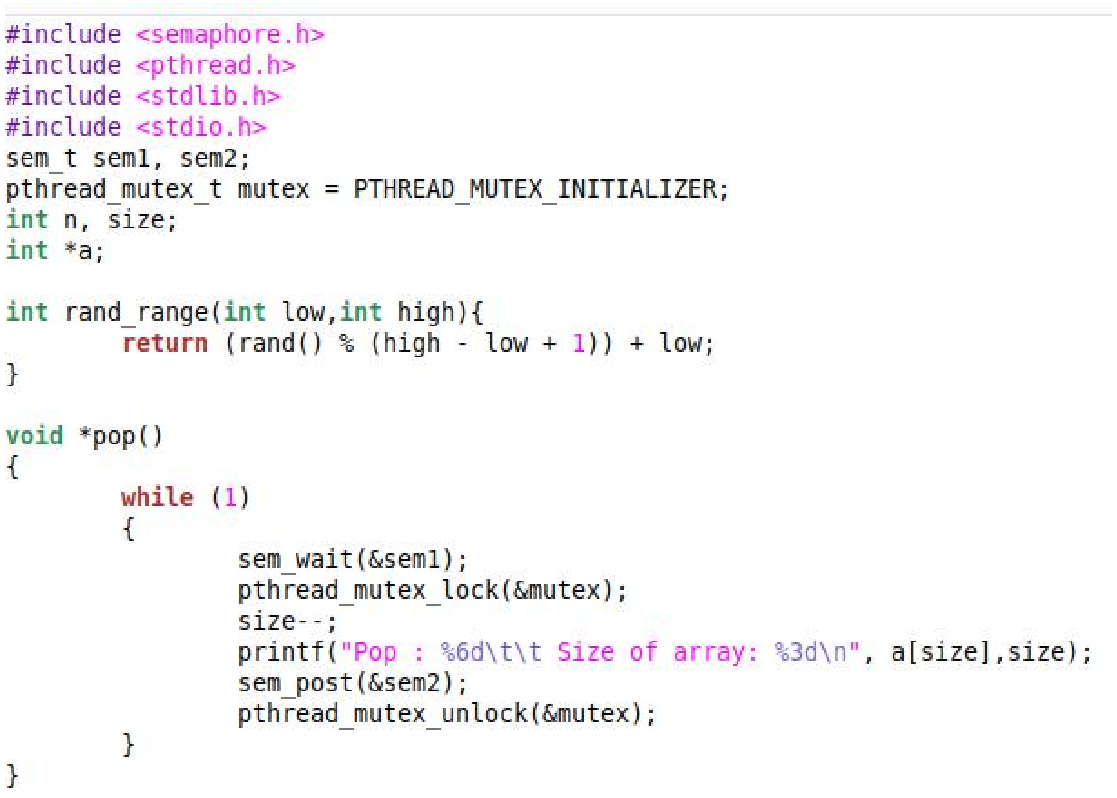
Demo:



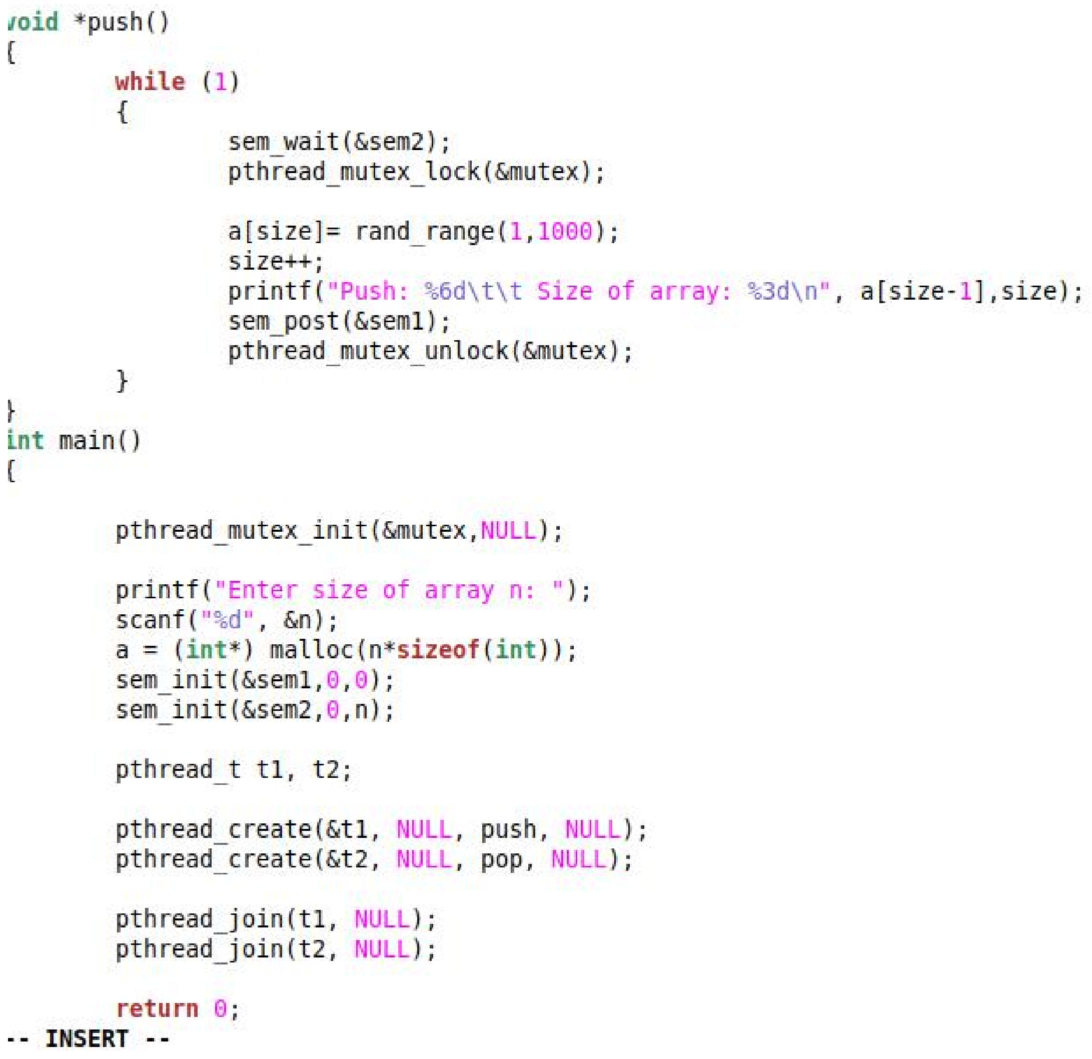
```
products: sells = 6714, products = 6796, 82
products: sells = 6723, products = 6797, 74
products: sells = 6723, products = 6798, 75
products: sells = 6723, products = 6799, 76
products: sells = 6723, products = 6800, 77
products: sells = 6723, products = 6801, 78
products: sells = 6723, products = 6802, 79
products: sells = 6723, products = 6803, 80
products: sells = 6723, products = 6804, 81
products: sells = 6723, products = 6805, 82
sells:    sells = 6723, products = 6796, 73
sells:    sells = 6724, products = 6805, 81
sells:    sells = 6725, products = 6806, 81
sells:    sells = 6726, products = 6806, 80
sells:    sells = 6727, products = 6806, 79
sells:    sells = 6728, products = 6806, 78
sells:    sells = 6729, products = 6806, 77
sells:    sells = 6730, products = 6806, 76
sells:    sells = 6731, products = 6806, 75
sells:    sells = 6732, products = 6806, 74
sells:    sells = 6733, products = 6806, 73
sells:    sells = 6734, products = 6806, 72
sells:    sells = 6735, products = 6806, 71
sells:    sells = 6736, products = 6806, 70
sells:    sells = 6737, products = 6806, 69
sells:    sells = 6738, products = 6806, 68
products: sells = 6724, products = 6806, 82
products: sells = 6739, products = 6807, 68
products: sells = 6739, products = 6808, 69
products: sells = 6739, products = 6809, 70
products: sells = 6739, products = 6810, 71
products: sells = 6739, products = 6811, 72
products: sells = 6739, products = 6812, 73
products: sells = 6739, products = 6813, 74
products: sells = 6739, products = 6814, 75
products: sells = 6739, products = 6815, 76
products: sells = 6739, products = 6816, 77
products: sells = 6739, products = 6817, 78
```

Câu 2:

Code:

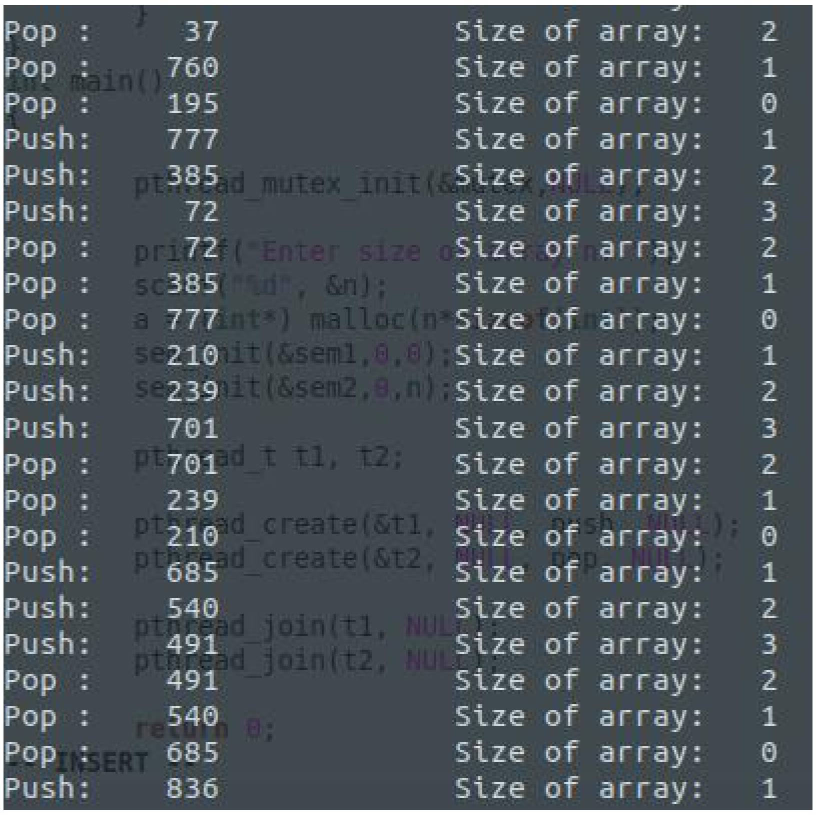


```c
#include <semaphore.h>
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
sem_t sem1, sem2;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int n, size;
int *a;

int rand_range(int low,int high){
        return (rand() % (high - low + 1)) + low;
}

void *pop()
{
        while (1)
        {
                sem_wait(&sem1);
                pthread_mutex_lock(&mutex);
                size--;
                printf("Pop : %6d\t\t Size of array: %3d\n", a[size],size);
                sem_post(&sem2);
                pthread_mutex_unlock(&mutex);
        }
}
```



```
void *push()
{
        while (1)
        {
                sem_wait(&sem2);
                pthread_mutex_lock(&mutex);

                a[size]= rand_range(1,1000);
                size++;
                printf("Push: %6d\t\t Size of array: %3d\n", a[size-1],size);
                sem_post(&sem1);
                pthread_mutex_unlock(&mutex);
        }
}
int main()
{

        pthread_mutex_init(&mutex,NULL);

        printf("Enter size of array n: ");
        scanf("%d", &n);
        a = (int*) malloc(n*sizeof(int));
        sem_init(&sem1,0,0);
        sem_init(&sem2,0,n);

        pthread_t t1, t2;

        pthread_create(&t1, NULL, push, NULL);
        pthread_create(&t2, NULL, pop, NULL);

        pthread_join(t1, NULL);
        pthread_join(t2, NULL);

        return 0;
-- INSERT --
```

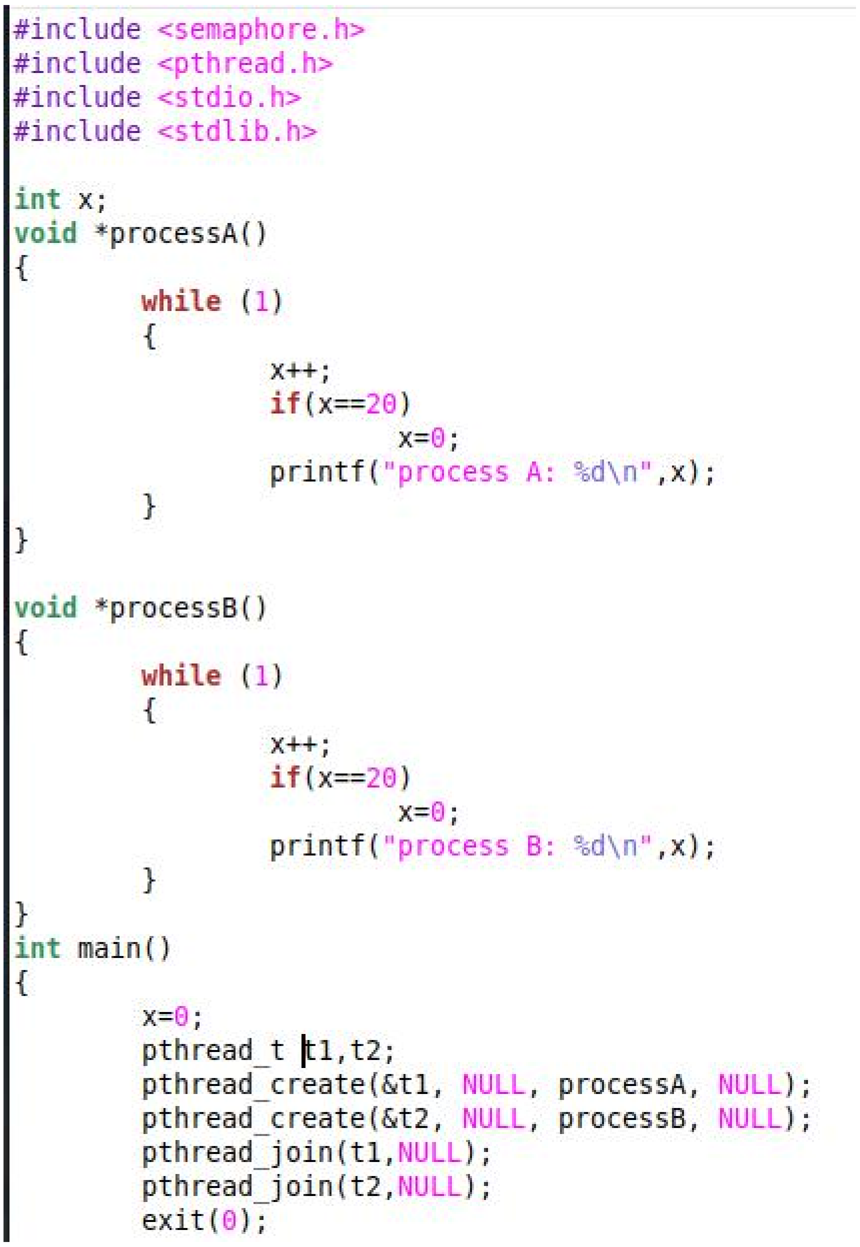
Demo:



```
Pop :    37          Size of array:   2
Pop :   760          Size of array:   1
Pop :   195          Size of array:   0
Push:   777          Size of array:   1
Push:   385          Size of array:   2
Push:    72          Size of array:   3
Pop :    72          Size of array:   2
Pop :   385          Size of array:   1
Pop :   777          Size of array:   0
Push:   210          Size of array:   1
Push:   239          Size of array:   2
Push:   701          Size of array:   3
Pop :   701          Size of array:   2
Pop :   239          Size of array:   1
Pop :   210          Size of array:   0
Push:   685          Size of array:   1
Push:   540          Size of array:   2
Push:   491          Size of array:   3
Pop :   491          Size of array:   2
Pop :   540          Size of array:   1
Pop :   685          Size of array:   0
Push:   836          Size of array:   1
```

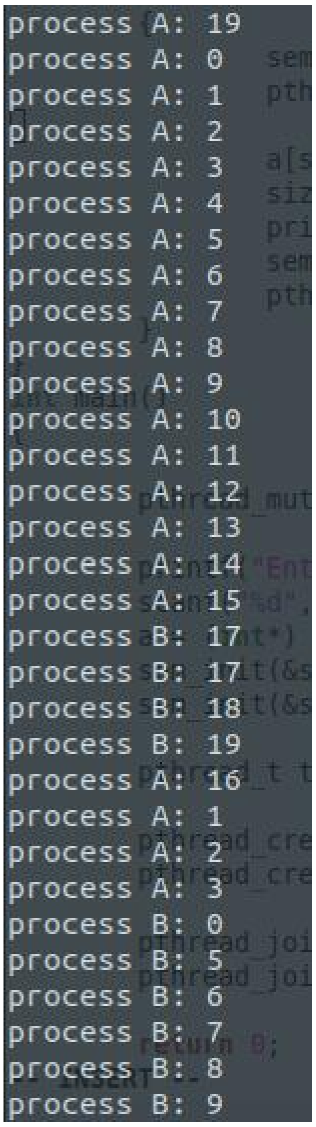
Câu 3:

Code:



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

int x;
void *processA()
{
        while (1)
        {
                x++;
                if(x==20)
                        x=0;
                printf("process A: %d\n",x);
        }
}

void *processB()
{
        while (1)
        {
                x++;
                if(x==20)
                        x=0;
                printf("process B: %d\n",x);
        }
}
int main()
{
        x=0;
        pthread_t t1,t2;
        pthread_create(&t1, NULL, processA, NULL);
        pthread_create(&t2, NULL, processB, NULL);
        pthread_join(t1,NULL);
        pthread_join(t2,NULL);
        exit(0);
```

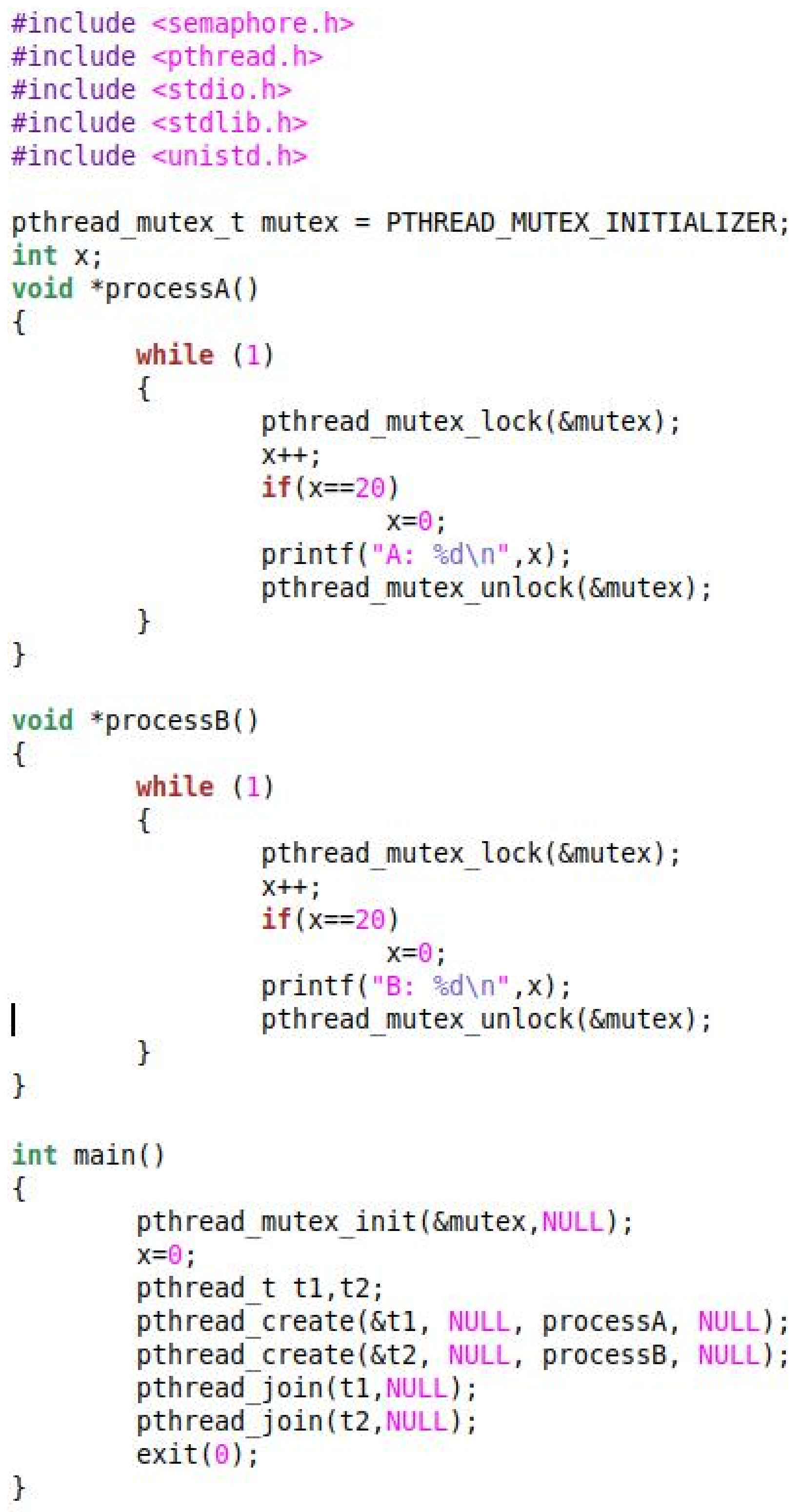
Demo:



```
process A: 19
process A: 0
process A: 1
process A: 2
process A: 3
process A: 4
process A: 5
process A: 6
process A: 7
process A: 8
process A: 9
process A: 10
process A: 11
process A: 12
process A: 13
process A: 14
process A: 15
process B: 17
process B: 17
process B: 18
process B: 19
process A: 16
process A: 1
process A: 2
process A: 3
process B: 0
process B: 5
process B: 6
process B: 7
process B: 8
process B: 9
```

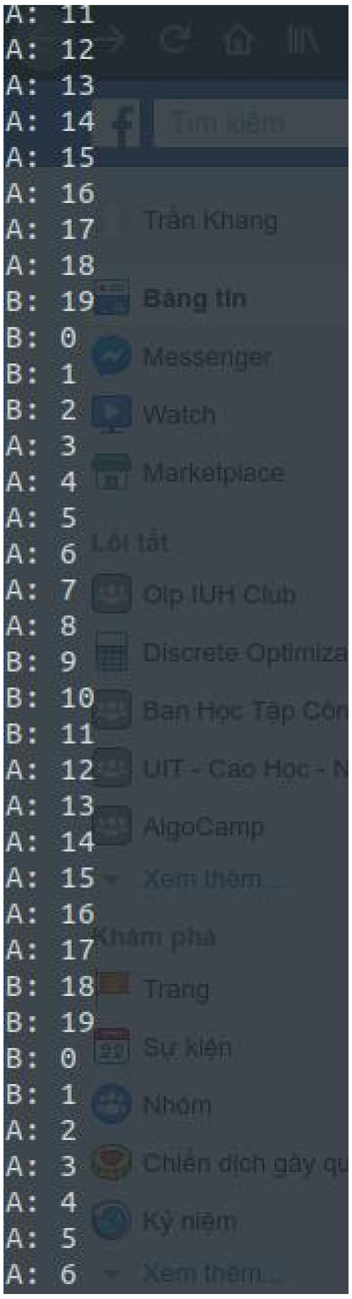
Câu 4:

Code:



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

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int x;
void *processA()
{
        while (1)
        {
                pthread_mutex_lock(&mutex);
                x++;
                if(x==20)
                        x=0;
                printf("A: %d\n",x);
                pthread_mutex_unlock(&mutex);
        }
}

void *processB()
{
        while (1)
        {
                pthread_mutex_lock(&mutex);
                x++;
                if(x==20)
                        x=0;
                printf("B: %d\n",x);
                pthread_mutex_unlock(&mutex);
        }
}

int main()
{
        pthread_mutex_init(&mutex,NULL);
        x=0;
        pthread_t t1,t2;
        pthread_create(&t1, NULL, processA, NULL);
        pthread_create(&t2, NULL, processB, NULL);
        pthread_join(t1,NULL);
        pthread_join(t2,NULL);
        exit(0);
}
```

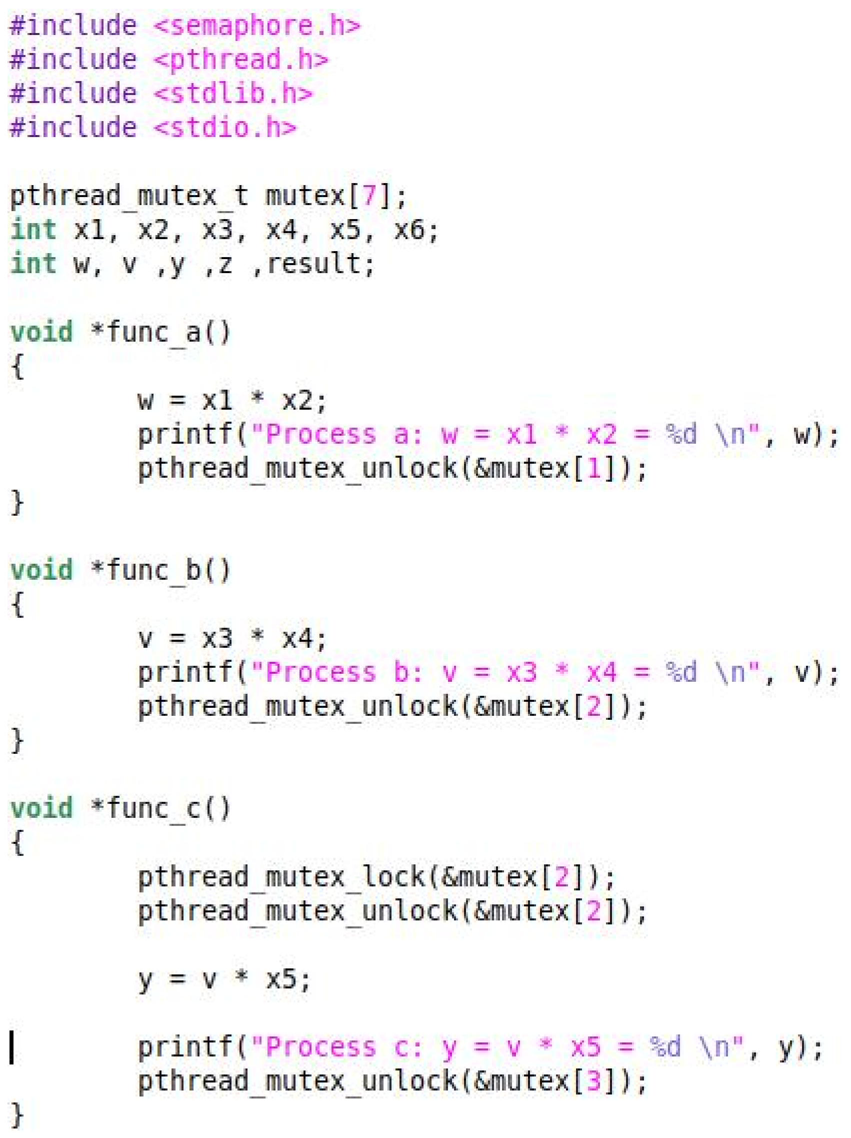
Demo:



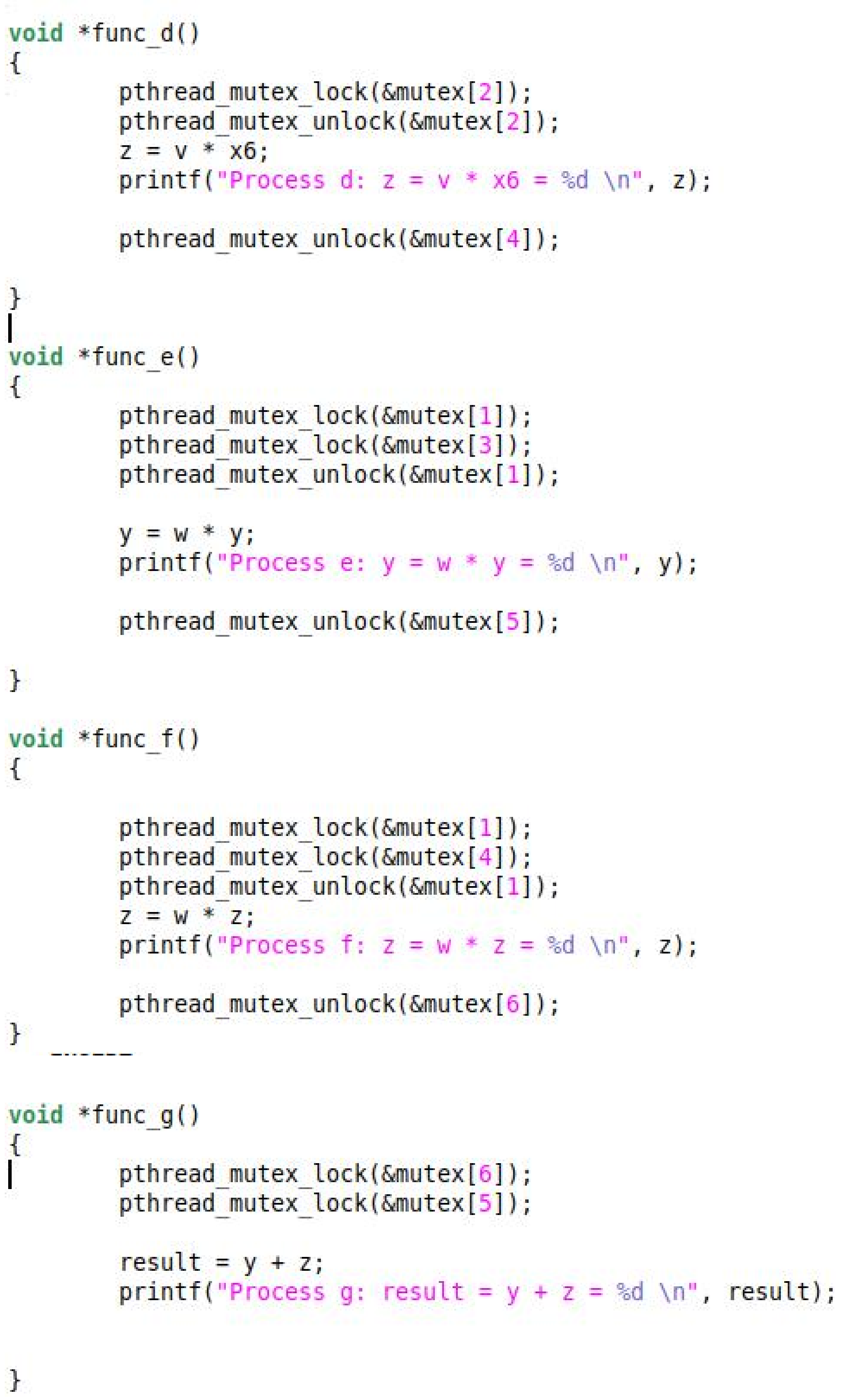
```
A: 11
A: 12
A: 13
A: 14
A: 15
A: 16
A: 17
A: 18
B: 19
B: 0
B: 1
B: 2
A: 3
A: 4
A: 5
A: 6
A: 7
A: 8
B: 9
B: 10
B: 11
A: 12
A: 13
A: 14
A: 15
A: 16
A: 17
B: 18
B: 19
B: 0
B: 1
A: 2
A: 3
A: 4
A: 5
A: 6
```

5.5 Bài Tập Ôn Tập

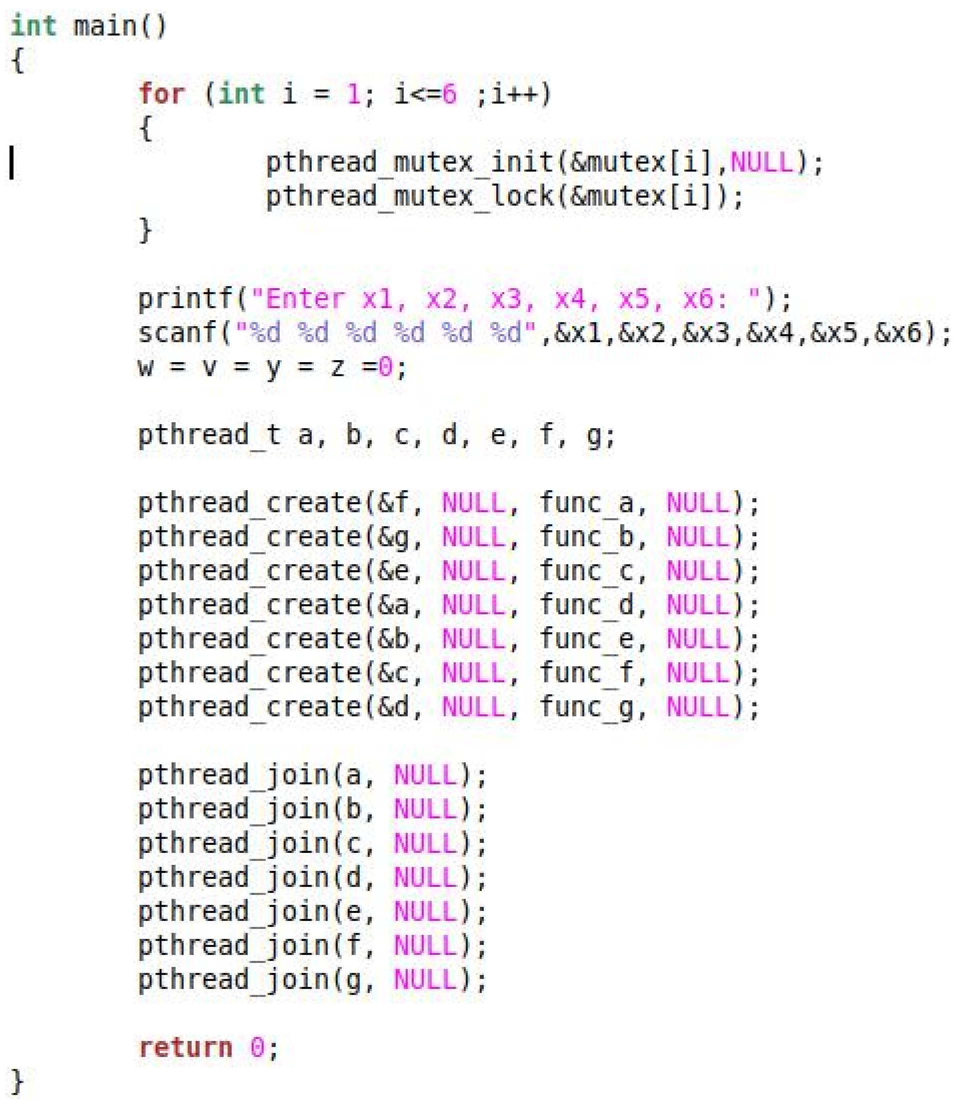
Code:



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

pthread_mutex_t mutex[7];
int x1, x2, x3, x4, x5, x6;
int w, v ,y ,z ,result;

void *func_a()
{
        w = x1 * x2;
        printf("Process a: w = x1 * x2 = %d \n", w);
        pthread_mutex_unlock(&mutex[1]);
}

void *func_b()
{
        v = x3 * x4;
        printf("Process b: v = x3 * x4 = %d \n", v);
        pthread_mutex_unlock(&mutex[2]);
}

void *func_c()
{
        pthread_mutex_lock(&mutex[2]);
        pthread_mutex_unlock(&mutex[2]);

        y = v * x5;

        printf("Process c: y = v * x5 = %d \n", y);
        pthread_mutex_unlock(&mutex[3]);
}
```

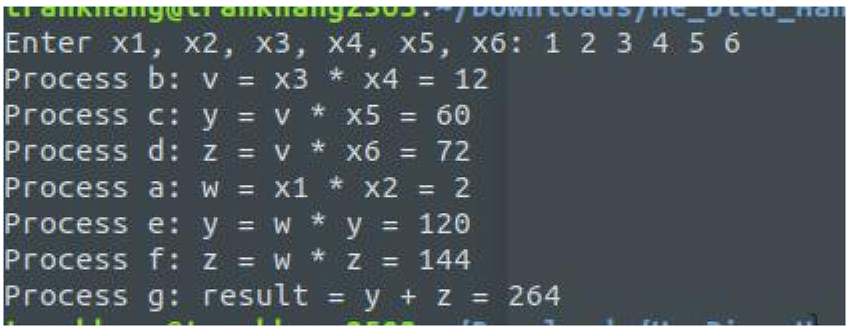


```
void *func_d()
{
        pthread_mutex_lock(&mutex[2]);
        pthread_mutex_unlock(&mutex[2]);
        z = v * x6;
        printf("Process d: z = v * x6 = %d \n", z);

        pthread_mutex_unlock(&mutex[4]);

}

void *func_e()
{
        pthread_mutex_lock(&mutex[1]);
        pthread_mutex_lock(&mutex[3]);
        pthread_mutex_unlock(&mutex[1]);

        y = w * y;
        printf("Process e: y = w * y = %d \n", y);

        pthread_mutex_unlock(&mutex[5]);

}

void *func_f()
{

        pthread_mutex_lock(&mutex[1]);
        pthread_mutex_lock(&mutex[4]);
        pthread_mutex_unlock(&mutex[1]);
        z = w * z;
        printf("Process f: z = w * z = %d \n", z);

        pthread_mutex_unlock(&mutex[6]);
}

void *func_g()
{
        pthread_mutex_lock(&mutex[6]);
        pthread_mutex_lock(&mutex[5]);

        result = y + z;
        printf("Process g: result = y + z = %d \n", result);


}
```



```
int main()
{
        for (int i = 1; i<=6 ;i++)
        {
                pthread_mutex_init(&mutex[i],NULL);
                pthread_mutex_lock(&mutex[i]);
        }

        printf("Enter x1, x2, x3, x4, x5, x6: ");
        scanf("%d %d %d %d %d %d",&x1,&x2,&x3,&x4,&x5,&x6);
        w = v = y = z =0;

        pthread_t a, b, c, d, e, f, g;

        pthread_create(&f, NULL, func_a, NULL);
        pthread_create(&g, NULL, func_b, NULL);
        pthread_create(&e, NULL, func_c, NULL);
        pthread_create(&a, NULL, func_d, NULL);
        pthread_create(&b, NULL, func_e, NULL);
        pthread_create(&c, NULL, func_f, NULL);
        pthread_create(&d, NULL, func_g, NULL);

        pthread_join(a, NULL);
        pthread_join(b, NULL);
        pthread_join(c, NULL);
        pthread_join(d, NULL);
        pthread_join(e, NULL);
        pthread_join(f, NULL);
        pthread_join(g, NULL);

        return 0;
}
```

Demo:



```
Enter x1, x2, x3, x4, x5, x6: 1 2 3 4 5 6
Process b: v = x3 * x4 = 12
Process c: y = v * x5 = 60
Process d: z = v * x6 = 72
Process a: w = x1 * x2 = 2
Process e: y = w * y = 120
Process f: z = w * z = 144
Process g: result = y + z = 264
```