```
/*----------------------------------------
UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3086 - Programacion de Microprocesadores
Modificado por: Kimberly Barrera
Fecha: 08/11/2020
Mod:   08/19/2022

retorno.cpp
Comparte desde subrutina argumento puntero
de tipo void, por medio de return.
----------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>

float resp = 0; // variable global debería estar protegida.

void *calculos(void *argument)
{
        float *input;
        input = (float *)argument;
        // debería iniciar el bloqueo con una variable mutex
        // Se utiliza memoria de heap para asegurar que el espacio se
memoria no se pierda cuando esta
        // función retorne y el hilo termine
        float res0 = 1/((*input)*(*input+1));
        resp = res0 + resp;

        float *output = (float *)malloc(sizeof(float));
        *output =  resp; //1/((*input)*(*input+1));
        // debería terminar el bloqueo con una variable mutex
        // Se convierte de float* a void* para el retorno
        return (void *)output;
}

int main()
{
        printf("\n\n");
        pthread_t thread_id;
        pthread_attr_t attr;

        pthread_attr_init(&attr);
        pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);

        // Donde se guardará el puntero de retorno de cada hilo
        void *exit_value;
```

```
        for (int i = 1; i < 10000; i++)
        {
                float v = (float)i;
                pthread_create(&thread_id, &attr, calculos, (void
*)&v);

                // Se unifica el thread, el cual nos dara su
resultado en `exit_value`
                pthread_join(thread_id, &exit_value); // Esperar a
que el hilo termine
        // Debería sacarlo del ciclo for para aumentar el nivel de
paralelísmo.

                // Se convierte de regreso de void* a float*
                float *result = (float *)exit_value;
                float res =*result;
                printf("Resultado es: %f\n", res);
                // Se debe liberar la memoria que se adquirió en el
hilo
                free(exit_value);
        }

        pthread_attr_destroy(&attr);
        pthread_exit(NULL);
        return 0;
}
```