Programs on thread

Thread creation1

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *add(void *arg){
printf("Thread called function stmt\n");
return NULL;}

void main(){
pthread_t tid;
pthread_create(&tid, NULL, add, NULL);
pthread_join(tid, NULL); }
```

Thread creation2

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *hello(){
printf("Welcome Everyone\n");}

void main(){
pthread_t tid;
pthread_create(&tid, NULL, hello, NULL);
pthread_join(tid, NULL);}
```

Thread creation3

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *sum(void *val){
int *val_num = (int *)(val);
printf("Value = %d\n", *val_num);}

void main(){
int val = 5;
pthread_t tid;
pthread_create(&tid, NULL, sum, (void *)&val);
pthread_join(tid, NULL);}
```

program 4

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

```
#include <pthread.h>

// Function to be executed by each thread
void *thread_function(void *arg) {
    int *thread_id = (int *)arg;
    printf("Thread %d: Hello, I am a new thread!\n", *thread_id);
    return NULL; }

int main() {
    pthread_t threads[5];  // Array to hold thread identifiers
    int thread_ids[5];     // Array to pass unique IDs to threads

    // Create 5 threads
    for (int i = 0; i < 5; i++) {
        thread_ids[i] = i + 1;
        if (pthread_create(&threads[i], NULL, thread_function, &thread_ids[i]) != 0) {
            perror("Failed to create thread");
            return 1; } }

    // Wait for all threads to complete
    for (int i = 0; i < 5; i++) {
        if (pthread_join(threads[i], NULL) != 0) {
            perror("Failed to join thread");
            return 1;  } }
    printf("All threads have completed execution.\n");
    return 0; }
```

Program 5

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

// Function to be executed by each thread
void *thread_function(void *arg) {
    int *thread_id = (int *)arg;
    printf("Thread %d: Executing...\n", *thread_id);
    int *result = malloc(sizeof(int));
    *result = *thread_id * 10; // Example computation
    pthread_exit(result);  }    // Return result to the joining thread


int main() {
    pthread_t threads[3];  // Array to hold thread identifiers
    int thread_ids[3];     // Array to pass unique IDs to threads
    // Create threads
    for (int i = 0; i < 3; i++) {
        thread_ids[i] = i + 1;
        if (pthread_create(&threads[i], NULL, thread_function, &thread_ids[i]) != 0) {
            perror("Failed to create thread");
```

```
        return 1;  } }

// Join threads and collect their results
for (int i = 0; i < 3; i++) {
    int *result;
    if (pthread_join(threads[i], (void **)&result) != 0) {
        perror("Failed to join thread");
        return 1;
    }
    printf("Thread %d joined. Result: %d\n", thread_ids[i], *result);
    free(result); // Free memory allocated by the thread
}

printf("All threads have completed execution.\n");
return 0;}
```