1.To find sum of an array

```
#include <iostream>
#include <pthread.h>
#define MAX 16
#define MAX_THREAD 4

using namespace std;

int a[] = { 1, 5, 7, 10, 12, 14, 15, 18, 20, 22, 25, 27, 30, 64, 110, 500 };
int sum[4] = { 0 };
int part = 0;

void* sum_array(void* arg)
{

    int thread_part = part++;

    for (int i = thread_part * (MAX / 4); i < (thread_part + 1) * (MAX / 4);
i++)
        sum[thread_part] += a[i];
}

int main()
{

    pthread_t threads[MAX_THREAD];

    for (int i = 0; i < MAX_THREAD; i++)
        pthread_create(&threads[i], NULL, sum_array, (void*)NULL);

    for (int i = 0; i < MAX_THREAD; i++)
        pthread_join(threads[i], NULL);

    int total_sum = 0;
    for (int i = 0; i < MAX_THREAD; i++)
        total_sum += sum[i];

    cout << "sum is " << total_sum << endl;

    return 0;
}
```

2.To search for an element in an array

```
#include <iostream>
#include <pthread.h>
```

```
using namespace std;

#define max 16
#define thread_max 4

int a[max] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

int key = 6;

int flag = 0;

int current_thread = 0;

void *threadSearch (void* args) {
    int num = current_thread++;
    for(int i=num*(max/4);i<(num+1)*(max/4);i++) {
        if(a[i] == key) flag = 1;
    }
}

int main() {
    pthread_t thread[thread_max];
    for(int i=0;i<thread_max;i++) {
        pthread_create(&thread[i],NULL,threadSearch,(void *)NULL);
    }
    for(int i=0;i<thread_max;i++) {
        pthread_join(thread[i],NULL);
    }
    if(flag == 1) {
        cout<<"Found"<< endl;
    }
    else cout<<"Not found" << endl;
    return 0;
}
```

3.To find maximum element of an array

```
#include <iostream>
#include <pthread.h>
using namespace std;

#define max 16
#define thread_max 4

int a[max] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

int max_num[thread_max] = {0};
```

```
int thread_num = 0;

void* findMax(void* args) {
    int num = thread_num++;
    int maxs = 0;
    for(int i=num*(max/4);i<(num+1)*(max/4);i++) {
        if(a[i]>maxs) {
            maxs = a[i];
        }
    }
    max_num[num] = maxs;
}

int main() {
    int maxs = 0;
    pthread_t thread[thread_max];
    for(int i=0;i<thread_max;i++) {
        pthread_create(&thread[i],NULL,findMax,(void*)NULL);
    }
    for(int i=0;i<thread_max;i++) {
        pthread_join(thread[i],NULL);
    }
    for(int i=0;i<thread_max;i++) {
        if(max_num[i]>maxs) {
            maxs = max_num[i];
        }
    }
    cout<< maxs << endl;
    return 0;
}
```

4.To perform addition and subtraction of matrices

```
#include <iostream>
#include <thread>

using namespace std;

// Size of Matrix
const int MAX = 4;

// Maximum number of threads
const int MAX_THREAD = 4;

// matAdd and matSub to store results
int matAdd[MAX][MAX];
```

```
int matSub[MAX][MAX];
int step_add = 0, step_sub = 0;

// Function to print matrix in readable format
void printMatrix(int mat[][MAX])
{
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}

// Function to perform matrix subtraction
void Subtraction(int matA[][MAX], int matB[][MAX])
{
    int index = step_sub++;

    // Each thread computes 1/4th of matrix addition
    int start = index * (MAX / 4);
    int end = (index + 1) * (MAX / 4);
    for (int i = start; i < end; i++) {
        for (int j = 0; j < MAX; j++) {
            matSub[i][j] = matA[i][j] - matB[i][j];
        }
    }
}

// Function to perform matrix Addition
void Addition(int matA[][MAX], int matB[][MAX])
{
    int index = step_add++;

    // Each thread computes 1/4th of matrix addition
    int start = index * (MAX / 4);
    int end = (index + 1) * (MAX / 4);
    for (int i = start; i < end; i++) {
        for (int j = 0; j < MAX; j++) {
            matAdd[i][j] = matA[i][j] + matB[i][j];
        }
    }
}

int main()
{
```

```
    // matrix A used for muliplication
    int matA[MAX][MAX] = { { 3, 7, 3, 6 },
                           { 9, 2, 0, 3 },
                           { 0, 2, 1, 7 },
                           { 2, 2, 7, 9 } };

    // matrix B used for multiplication
    int matB[MAX][MAX] = { { 6, 5, 5, 2 },
                           { 1, 7, 9, 6 },
                           { 6, 6, 8, 9 },
                           { 0, 3, 5, 2 } };

    cout << "Matrix A:" << endl;
    printMatrix(matA);
    cout << "Matrix B:" << endl;
    printMatrix(matB);

    // Creating list of size MAX_THREAD
    thread add_thread[MAX_THREAD];
    thread sub_thread[MAX_THREAD];

    // Creating MAX_THEAD number of threads
    for (int i = 0; i < MAX_THREAD; i++) {
        add_thread[i] = thread(Addition, matA, matB);
        sub_thread[i] = thread(Subtraction, matA, matB);
    }

    // Waiting for all threads to finish
    for (int i = 0; i < MAX_THREAD; i++) {
        add_thread[i].join();
        sub_thread[i].join();
    }

    // Printing the resultant matrices
    cout << "Sum of Matrix A and B:" << endl;
    printMatrix(matAdd);
    cout << "Subtraction of Matrix A and B:" << endl;
    printMatrix(matSub);

    return 0;
}
```