Part B

Task#1

Perform problem is currently being solved using semaphore and threads. There are 3 threads for reader and 2 threads for writer that have been created.

```cpp
#include<semaphore.h>
#include<stdio.h>
#include<pthread.h>
# include<bits/stdc++.h>
using namespace std;

int readcount = 0;
int writecount = 0;
int bsize[5];

string performa = "This is performa";

sem_t x,y,z,rsem, wsem;
pthread_t r[3], w[2];

void *reader(void *args)
{

        sem_wait(&z);
        sem_wait(&rsem);
        sem_wait(&x);
        readcount++;

        if(readcount == 1)
            sem_wait(&wsem);

        sem_post(&x);
```

```cpp
        sem_post(&rsem);
        sem_post(&z);

        cout << "\nReading Val : " << performa << endl;
        sem_wait(&x);
        readcount--;

        if(readcount==0)
            sem_post(&wsem);

        sem_post(&x);

        return NULL;
}

void *writer(void *args)
{
        sem_wait(&y);
        writecount++;
        if(writecount==1)
                sem_wait(&rsem);

        sem_post(&y);
        sem_wait(&wsem);

        performa = performa + " Modified";
```

```
54          cout << "\nPerforma Modified Value is: " << performa << endl;
55
56          sem_post(&wsem);
57          sem_wait(&y);
58          writecount--;
59
60          if(writecount==0)
61                  sem_post(&rsem);
62
63          sem_post(&y);
64
65          return NULL;
66  }
67
68  int main()
69  {
70          sem_init(&x,0,1);
71          sem_init(&wsem,0,1);
72          sem_init(&y,0,1);
73          sem_init(&z,0,1);
74          sem_init(&rsem,0,1);
75
76          pthread_create(&r[0],NULL, reader, NULL);
77          pthread_create(&w[0],NULL, writer, NULL);
78          pthread_create(&r[1],NULL, reader, NULL);
79          pthread_create(&w[1],NULL, writer, NULL);
80          // pthread_create(&r[2],NULL, reader, NULL);
```

```
80          // pthread_create(&r[2],NULL, reader, NULL);
81          // pthread_create(&r[3],NULL, reader, NULL);
82          // pthread_create(&r[4],NULL, reader, NULL);
83
84          pthread_join(r[0],NULL);
85          pthread_join(w[0],NULL);
86          pthread_join(r[1],NULL);
87          pthread_join(w[1],NULL);
88
89          // pthread_join(r[2],NULL);
90          // pthread_join(r[3],NULL);
91          // pthread_join(r[4],NULL);
92
93          return 0;
94  }
95
```

```
sam@sam-VirtualBox:~/Desktop$ g++ performa.cpp -o outp -pthread
sam@sam-VirtualBox:~/Desktop$ ./outp

Reading Val : This is performa

Performa Modified Value is: This is performa Modified

Reading Val : This is performa Modified

Performa Modified Value is: This is performa Modified Modified
sam@sam-VirtualBox:~/Desktop$
```

Task#2

An array of size 10 has been declared earlier for the insertion of values by respective threads. One thread inserts values from 1 – 5 and the next one does the same job of inserting from 6 – 10. Lasting main method is used for the calling and calculation of the rest of the process.

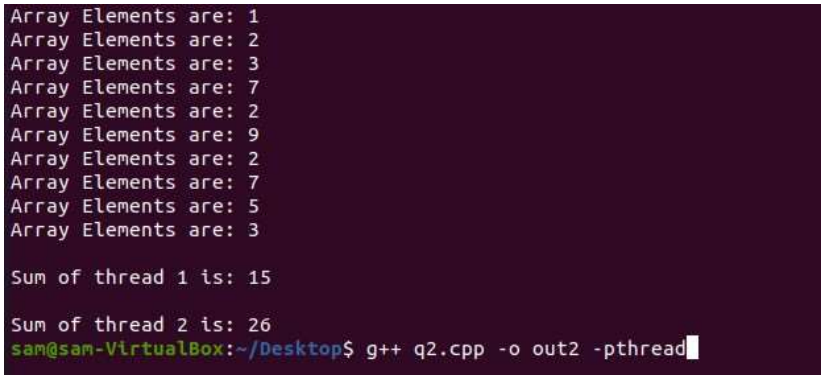
```cpp
#include <semaphore.h>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <pthread.h>
#include <thread>

using namespace std;

sem_t lk;
int arr[10];

void *thread_1(void* args){

        int val = 0;

        sem_wait(&lk);
        for (int i = 0; i < 5; i++){
                cout << "Enter the valeus one by one" << endl;
                cin >> val;
                arr[i] = val;
        }
        sem_post(&lk);

        return NULL;
```

```cpp
}

void *thread_2(void* args){

        cout << "" << endl;

        int val = 0;

        sem_wait(&lk);
        for (int i = 5; i < 10; i++){
                cout << "Enter the valeus one by one" << endl;
                cin >> val;
                arr[i] = val;
        }
        sem_post(&lk);

        return NULL;
}


int main(){

        sem_init(&lk, 0, 1);
        pthread_t t1, t2;

        pthread_create(&t1, NULL, thread_1, NULL);
```

```
54          pthread_create(&t2, NULL, thread_2, NULL);
55
56          pthread_join(t1, NULL);
57          pthread_join(t2, NULL);
58
59
60          for (int i = 0; i < 10; i++){
61
62                  cout << "Array Elements are: " << arr[i] << endl;
63          }
64
65          int sum_1 = 0;
66          int sum_2 = 0;
67
68          cout << "" << endl;
69
70          for (int i = 0; i < 5; i++){
71
72                  sum_1 += arr[i];
73          }
74          cout << "Sum of thread 1 is: " << sum_1 <<endl;
75
76
77          cout << "" << endl;
78
79          for (int i = 5; i < 10; i++){
80
```

```
81                  sum_2 += arr[i];
82            }
83            cout << "Sum of thread 2 is: " << sum_2 <<endl;
84
85
86            return 0;
87
88  }
```

```
sam@sam-VirtualBox:~/Desktop$ g++ q2.cpp -o out2 -pthread
sam@sam-VirtualBox:~/Desktop$ ./out2
Enter the valeus one by one

1
Enter the valeus one by one
2
Enter the valeus one by one
3
Enter the valeus one by one
7
Enter the valeus one by one
2
Enter the valeus one by one
9
Enter the valeus one by one
2
Enter the valeus one by one
7
Enter the valeus one by one
5
Enter the valeus one by one
3
```


Array Elements are: 1
Array Elements are: 2
Array Elements are: 3
Array Elements are: 7
Array Elements are: 2
Array Elements are: 9
Array Elements are: 2
Array Elements are: 7
Array Elements are: 5
Array Elements are: 3
Sum of thread 1 is: 15
Sum of thread 2 is: 26
sam@sam-VirtualBox:~/Desktop$ g++ q2.cpp -o out2 -pthread