1. Steps of how a user can pass their tasks, frames, and frame time to your framework:
First, specify the tasks using struct task_item structures similar to what shown below:

```
    struct task_item t0;
    struct task_item t1;
    struct task_item t2;
    struct task_item t3;
    struct task_item t4;
    t0=(struct task_item){
    .type = 1, //0 for slack, 1 for periodic,2 for sporadic, 3 for aperiodic
    .WCET = 20, //WCET in ms
    .ddl = 600,//relative deadline in ms
    .min_arr = 0, //minumum interarrival time for sporadic tasks
    .task=&engine_low_pressure_compression_shaft_task, //pointer to task's
function
    };
    t1=(struct task_item){
    .type = 1, //0 for slack, 1 for periodic,2 for sporadic, 3 for aperiodic
    .WCET = 20, //WCET in ms
    .ddl = 300,//relative deadline in ms
    .min_arr = 0, //minumum interarrival time for sporadic tasks
    .task=&engine_high_pressure_compression_shaft_task, //pointer to task's
function
    };
    t2=(struct task_item){
    .type = 1, //0 for slack, 1 for periodic,2 for sporadic, 3 for aperiodic
    .WCET = 10, //WCET in ms
    .ddl = 150,//relative deadline in ms
    .min_arr = 0, //minumum interarrival time for sporadic tasks
    .task=&engine_fuel_injection_task, //pointer to task's function
    };
    t3=(struct task_item){
    .type = 2, //0 for slack, 1 for periodic,2 for sporadic, 3 for aperiodic
    .WCET = 50, //WCET in ms
    .ddl = 200,//relative deadline in ms
    .min_arr = 400, //minumum interarrival time for sporadic tasks
    .task=&engine_airflow_monitoring_task, //pointer to task's function
    };
    t4=(struct task_item){
    .type = 3, //0 for slack, 1 for periodic,2 for sporadic, 3 for aperiodic
    .WCET = 30, //WCET in ms
    .ddl = 500,//relative deadline in ms
    .min_arr = 0, //minumum interarrival time for sporadic tasks
```

```
    .task=&engine_vibration_analysis_task, //pointer to task's function
    };
```

Then, create an array of pointers of type *struct task_item* * and pass in the previously declared task items by referencing:

```
    struct task_item *test_a[5];
    test_a[0]=&t0;
    test_a[1]=&t1;
    test_a[2]=&t2;
    test_a[3]=&t3;
    test_a[4]=&t4;
```

Then, create each frame in the sequence of frames as arrays of int type, where an item in an array is either:
1. a non-negative number representing a periodic task at the regarding index in the array of tasks previously created;
2. a negative number representing a slack time with the duration of its absolute value (in ms). E.g. "-70" means a slack time with duration of 70ms.

```
    int test_b_0[3] = {0,2,-70};
    int test_b_1[3] = {1,2,-70};
    int test_b_2[2] = {2,-90};
    int test_b_3[2] = {2,-90};
    int test_b_4[3] = {1,2,-70};
    int test_b_5[2] = {2,-90};
```

Now put these frames into an array of type *int* * to form a 2D array representing the sequence of frames. Also create an unsigned int array showing the size of sub-array at respective index in the 2D array just created. E.g. test_c[0]=3 means test_b[0] has 3 items.

```
    int *test_b[6] = {test_b_0,test_b_1,test_b_2,test_b_3,test_b_4,test_b_5};
    unsigned int test_c[6] = {3,3,2,2,3,2};
```

Finally, pass the objects created above into the framework in the following order:
*The last parameter is the length of each frame in ms.

```
    CyclicExecutionFramework(5, test_a, 6, test_b, test_c, 100);
```

2. Steps of how the user's job code gets called by the framework code and where it should be written:
The periodic tasks will be called as indicated in the sequence of frames. And for each slack time, a pre-empted periodic task/sporadic task/aperiodic task will be run in its duration depending on the exact situation. The code (function) are passed into

the framework through the function pointers in the task field of each *struct task_item* structure.

3. A list of your chosen frames and frame time from part 2, and the reasoning behind the chosen frame schedules and frame time:

```
int test_b_0[3] = {0,2,-70};
int test_b_1[3] = {1,2,-70};
int test_b_2[2] = {2,-90};
int test_b_3[2] = {2,-90};
int test_b_4[3] = {1,2,-70};
int test_b_5[2] = {2,-90};
```

The sequence of frames are shown as above, where:

0 refers to engine_low_pressure_compression_shaft_task;

1 refers to engine_high_pressure_compression_shaft_task;

2 refers to engine_fuel_injection_task;

It is noticeable that the periodic tasks are put consecutively at the beginning of each frame to maximize their success probability and create longest possible slack times.

A negative number refers to a slack time with duration in ms of its absolute value.

While the frame time is 100 ms because:

1. 100ms > max (WCET_of_tasks) = 50ms;
2. the hyperperiod, 600ms, is evenly divided by 100ms;
3. $2f$ – gcd(Pi, f) $\leq D$ is satisfied because $2f$ – gcd(Pi, f)=2*100-100=100 while min(D)=150.