IMPLEMENTATION OF TASK QUEUES FOR SCHEDULER

ASSIGNMENT-01

BASATI SIVAKRISHNA

22976

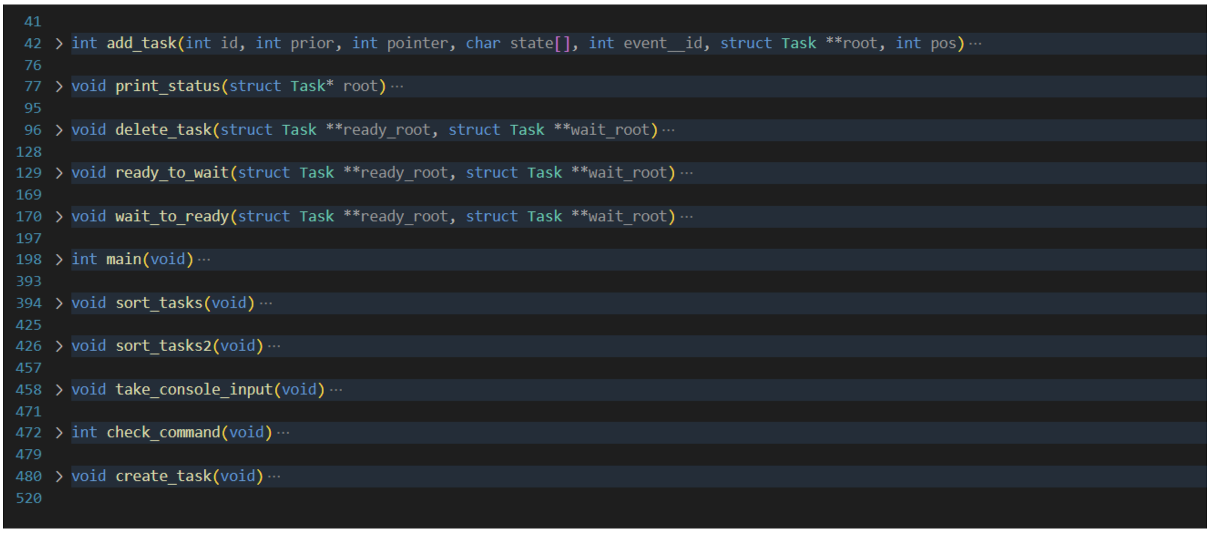
DESE(EPD)

Initial tasks are placed with all details in example1.txt file and read by the code once it starts running.

CODE OVERVIEW:

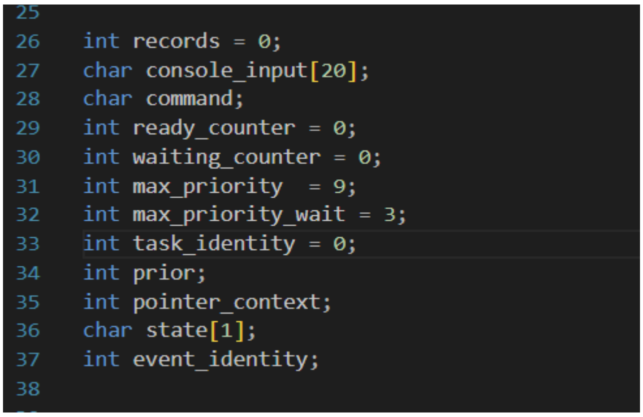
1. **FUNCTIONS:**

These are all the functions used for task scheduling.

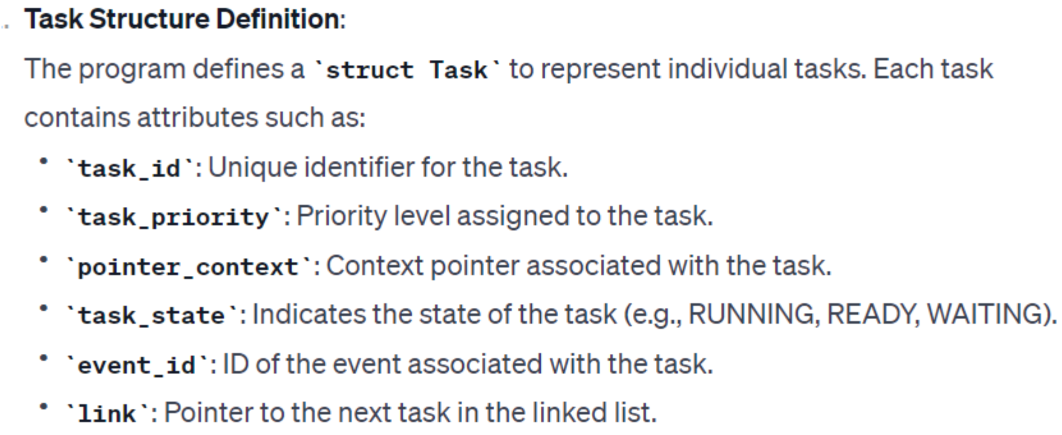


```
int add_task(int id, int prior, int pointer, char state[], int event__id, struct Task **root, int pos) ···

void print_status(struct Task* root) ···

void delete_task(struct Task **ready_root, struct Task **wait_root) ···

void ready_to_wait(struct Task **ready_root, struct Task **wait_root) ···

void wait_to_ready(struct Task **ready_root, struct Task **wait_root) ···

int main(void) ···

void sort_tasks(void) ···

void sort_tasks2(void) ···

void take_console_input(void) ···

int check_command(void) ···

void create_task(void) ···
```

2. **MACROS:**



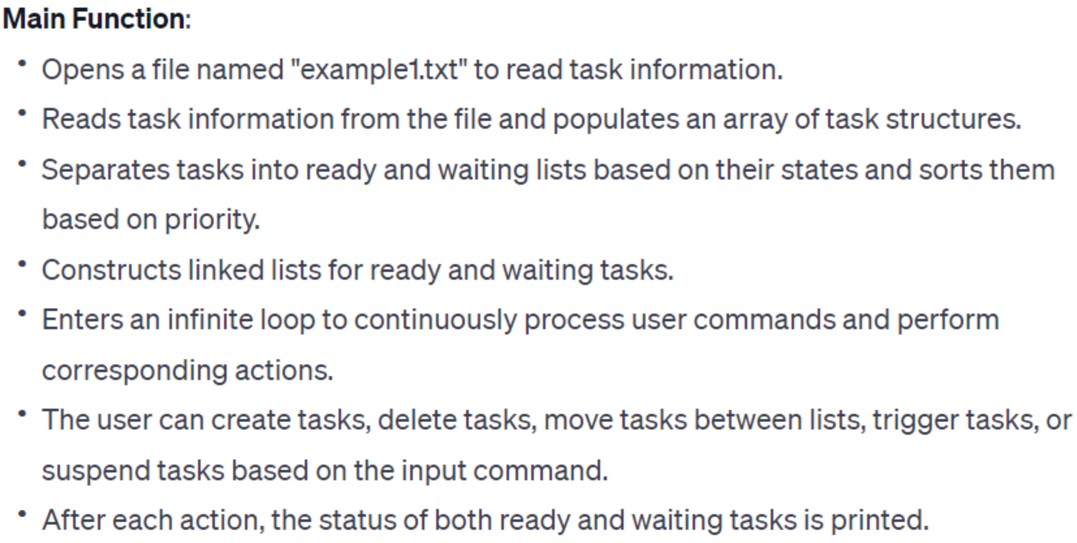
```
int records = 0;
char console_input[20];
char command;
int ready_counter = 0;
int waiting_counter = 0;
int max_priority   = 9;
int max_priority_wait = 3;
int task_identity = 0;
int prior;
int pointer_context;
char state[1];
int event_identity;
```



Task Structure Definition:

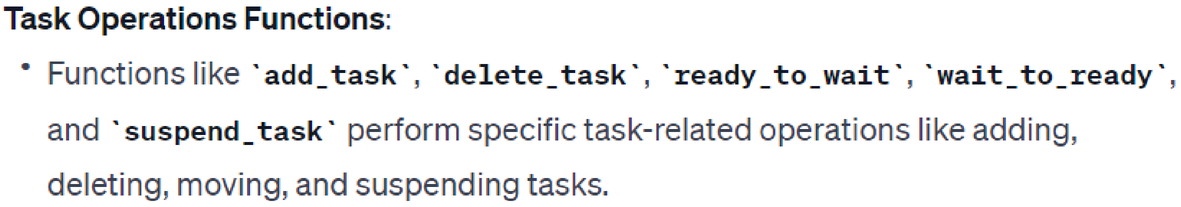
The program defines a `struct Task` to represent individual tasks. Each task contains attributes such as:

- `task_id`: Unique identifier for the task.
- `task_priority`: Priority level assigned to the task.
- `pointer_context`: Context pointer associated with the task.
- `task_state`: Indicates the state of the task (e.g., RUNNING, READY, WAITING).
- `event_id`: ID of the event associated with the task.
- `link`: Pointer to the next task in the linked list.



Main Function:

- Opens a file named "example1.txt" to read task information.
- Reads task information from the file and populates an array of task structures.
- Separates tasks into ready and waiting lists based on their states and sorts them based on priority.
- Constructs linked lists for ready and waiting tasks.
- Enters an infinite loop to continuously process user commands and perform corresponding actions.
- The user can create tasks, delete tasks, move tasks between lists, trigger tasks, or suspend tasks based on the input command.
- After each action, the status of both ready and waiting tasks is printed.

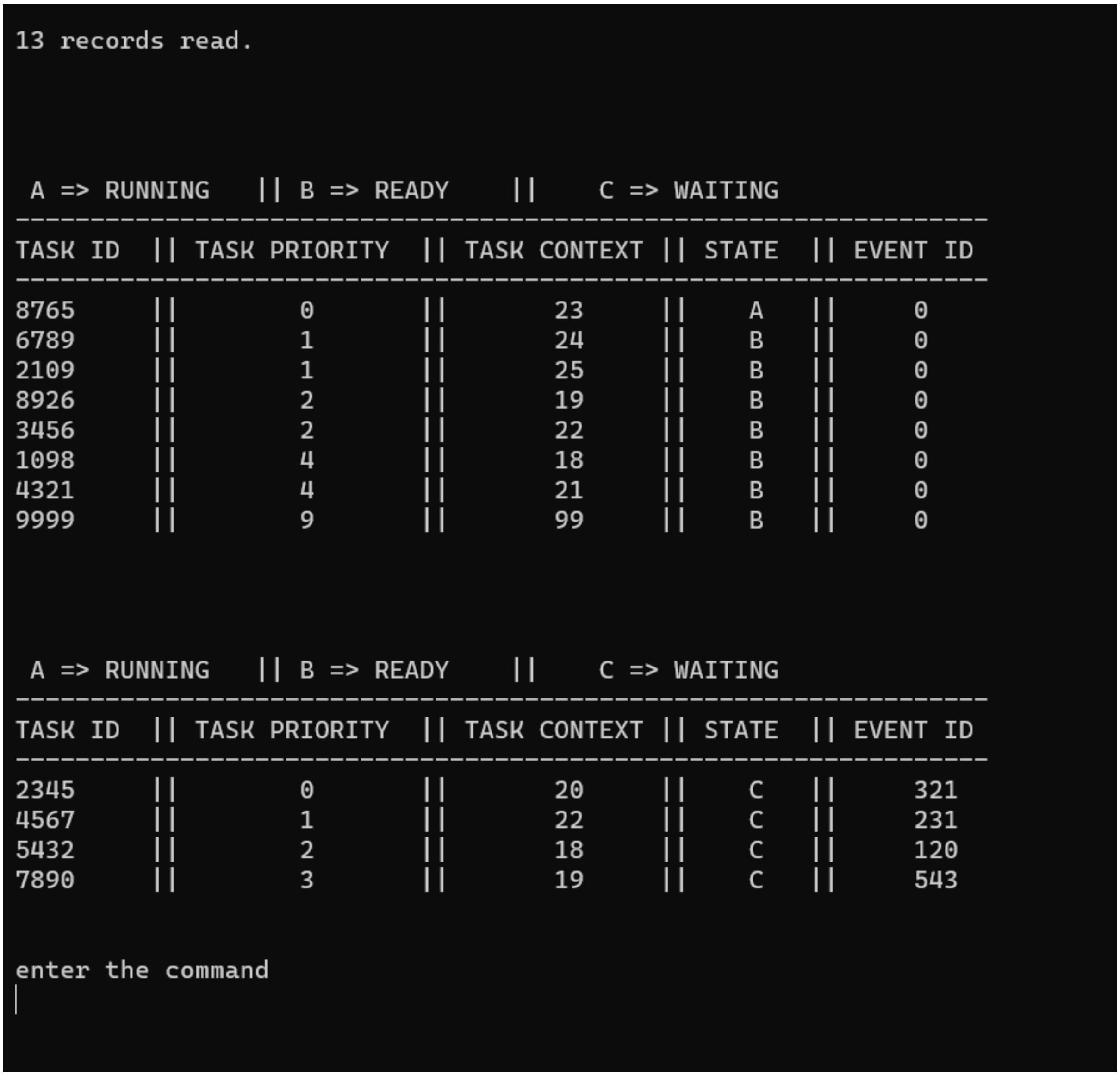


Task Operations Functions:

- Functions like `add_task`, `delete_task`, `ready_to_wait`, `wait_to_ready`, and `suspend_task` perform specific task-related operations like adding, deleting, moving, and suspending tasks.

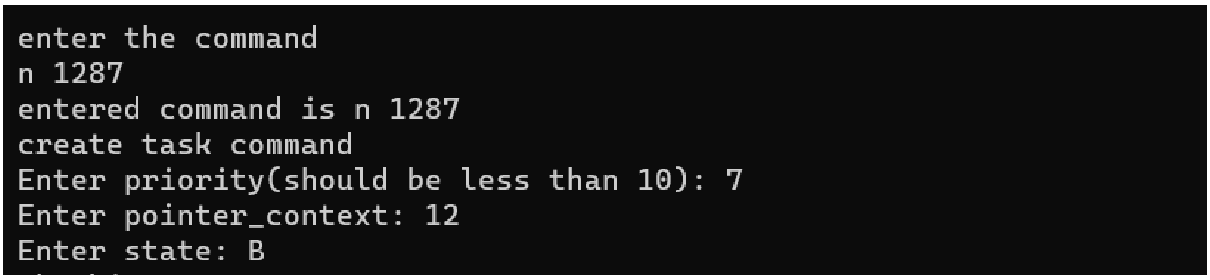
RESULTS:

Before giving any command

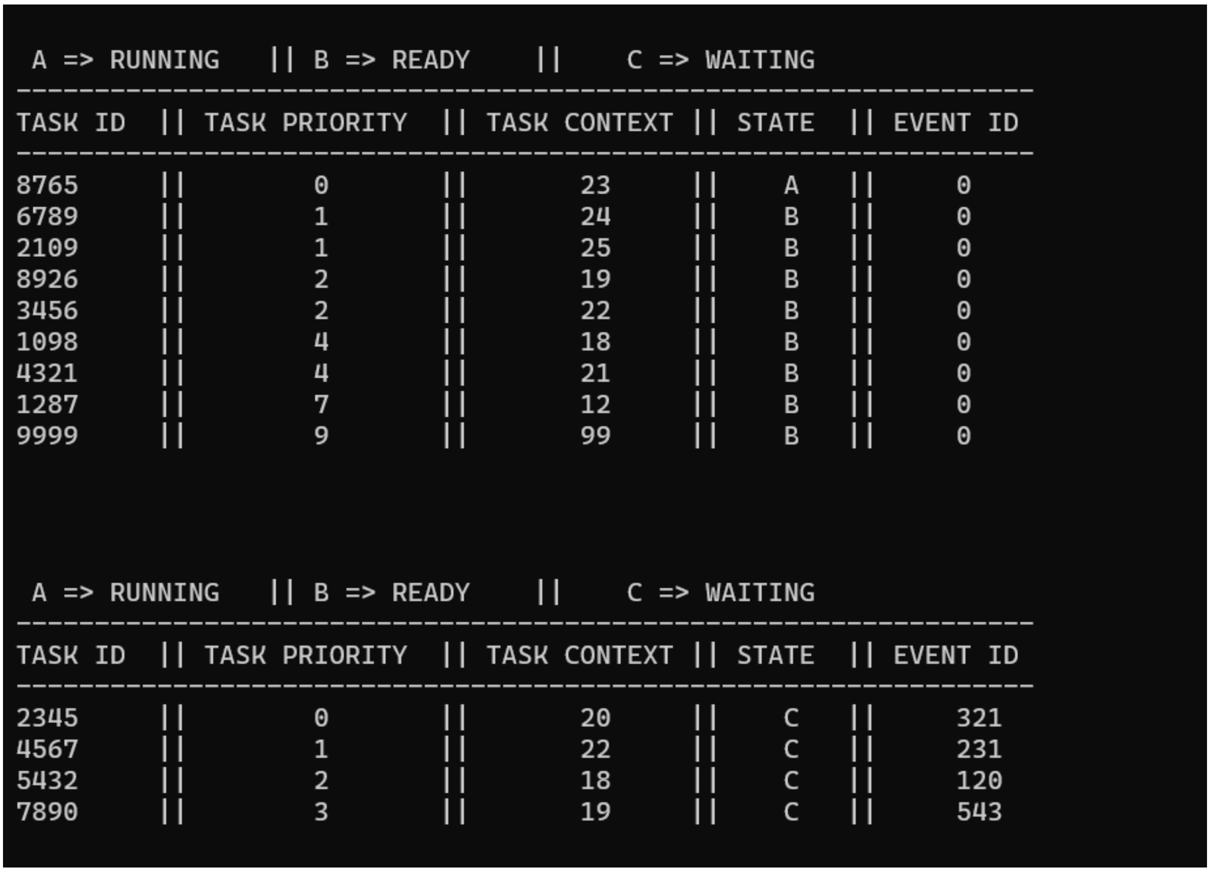


```
13 records read.



 A => RUNNING   || B => READY    ||    C => WAITING
-----------------------------------------------------------------
TASK ID   || TASK PRIORITY  || TASK CONTEXT || STATE   || EVENT ID
-----------------------------------------------------------------
8765      ||        0       ||       23     ||    A    ||     0
6789      ||        1       ||       24     ||    B    ||     0
2109      ||        1       ||       25     ||    B    ||     0
8926      ||        2       ||       19     ||    B    ||     0
3456      ||        2       ||       22     ||    B    ||     0
1098      ||        4       ||       18     ||    B    ||     0
4321      ||        4       ||       21     ||    B    ||     0
9999      ||        9       ||       99     ||    B    ||     0



 A => RUNNING   || B => READY    ||    C => WAITING
-----------------------------------------------------------------
TASK ID   || TASK PRIORITY  || TASK CONTEXT || STATE   || EVENT ID
-----------------------------------------------------------------
2345      ||        0       ||       20     ||    C    ||     321
4567      ||        1       ||       22     ||    C    ||     231
5432      ||        2       ||       18     ||    C    ||     120
7890      ||        3       ||       19     ||    C    ||     543


enter the command
```

After giving **n task_id** command:



```
enter the command
n 1287
entered command is n 1287
create task command
Enter priority(should be less than 10): 7
Enter pointer_context: 12
Enter state: B
```



```
 A => RUNNING   || B => READY    ||    C => WAITING
-----------------------------------------------------------------
TASK ID  || TASK PRIORITY  || TASK CONTEXT || STATE  || EVENT ID
-----------------------------------------------------------------
8765     ||         0       ||       23      ||     A   ||      0
6789     ||         1       ||       24      ||     B   ||      0
2109     ||         1       ||       25      ||     B   ||      0
8926     ||         2       ||       19      ||     B   ||      0
3456     ||         2       ||       22      ||     B   ||      0
1098     ||         4       ||       18      ||     B   ||      0
4321     ||         4       ||       21      ||     B   ||      0
1287     ||         7       ||       12      ||     B   ||      0
9999     ||         9       ||       99      ||     B   ||      0



 A => RUNNING   || B => READY    ||    C => WAITING
-----------------------------------------------------------------
TASK ID  || TASK PRIORITY  || TASK CONTEXT || STATE  || EVENT ID
-----------------------------------------------------------------
2345     ||         0       ||       20      ||     C   ||      321
4567     ||         1       ||       22      ||     C   ||      231
5432     ||         2       ||       18      ||     C   ||      120
7890     ||         3       ||       19      ||     C   ||      543
```

All operations checked accordingly.