#include <stdio.h>

#include <stdlib.h>

typedef struct process {

int id;

int timecycle;

struct process\* next;

} process;

process\* createprocess(int id, int timecycle) //create a new process node

{

process\* newprocess = (process\*)malloc(sizeof(process));

newprocess->id = id;

newprocess->timecycle = timecycle;

newprocess->next = NULL;

return newprocess;

}

process\* insertprocess(process\* head, int id, int timecycle) // Insert a process into the circular linked list

{

process\* newprocess = createprocess(id, timecycle);

if (head == NULL) // If the list is empty, make the new process the head

{

newprocess->next = newprocess;

return newprocess;

}

process\* current = head;

while (current->next != head) // Traverse to the end of the list

{

current = current->next;

}

current->next = newprocess;

newprocess->next = head;

return head;

}

process\* deleteprocess(process\* head, int id) // Delete a process into the circular linked list

{

if (head == NULL)

{

return NULL;

}

if (head->id == id)

{

if (head->next == head)

{

free(head);

return NULL;

}

process\* last = head;

while (last->next != head)

{

last = last->next;

}

last->next = head->next;

free(head);

return last->next;

}

process\* current = head;

process\* prev = NULL;

while (current->id != id)

{

if (current->next == head)

{

printf("Process with ID %d not found!\n", id);

return head;

}

prev = current;

current = current->next;

}

prev->next = current->next;

free(current);

return head;

}

void displayprocesses(process\* head, FILE\* file) // Display the processes in the circular linked list

{

if (head == NULL)

{

fprintf(file, "No processes in the list!\n");

return;

}

process\* current = head;

fprintf(file, "Process List: ");

do {

fprintf(file, "P%d:%d ", current->id, current->timecycle);

current = current->next;

} while (current != head);

fprintf(file, "\n");

}

void executeCycle(process\*\* headRef, FILE\* file) // execute the processes in the circular linked list

{

process\* head = \*headRef;

if (head == NULL)

{

fprintf(file, "No processes in the list!\n");

return;

}

process\* current = head;

do {

current->timecycle--;

if (current->timecycle == 0)

{

fprintf(file, "Executing process P%d\n", current->id);

\*headRef = deleteprocess(head, current->id);

head = \*headRef;

current = head;

}

else

{

current = current->next;

}

} while (current != head);

}

int main()

{

FILE\* output\_file = fopen("output.txt", "w");

if (output\_file == NULL)

{

printf("Unable to open the output file.\n");

return 1;

}

process\* head = NULL;

head = insertprocess(head, 1, 1); // Insert processes into the circular linked list

head = insertprocess(head, 2, 2);

head = insertprocess(head, 3, 3);

for (int i = 1; i <= 6; i++)

{ // Execute the cycles

fprintf(output\_file, "Cycle %d:\n", i);

executeCycle(&head, output\_file);

displayprocesses(head, output\_file);

fprintf(output\_file, "\n");

if (head == NULL)

{

fprintf(output\_file, "All processes executed!\n");

break;

}

}

fclose(output\_file);

process\* current = head; // Free the memory

while (current != NULL)

{

process\* temp = current;

current = current->next;

free(temp);

}

return 0;

}

Output



