SIMULATION-BASED PROJECT
REPORT

(Project TermAugust-December 2023)

Submitted by

Name of Student:- Sai Ganesh

Registration Number:- 12222126

Course Code:- CSE 316

Under the Guidanceof

Cherry Khosla : 13436
School of Computer Science and Engineering

Deceleration:-

I hereby declare that the project work entitled "Simulation-Based Project" is an authentic record of our own work carried out as requirements for the award of a B.Tech degree in Computer Science and Engineering (Data Science with

ML) from Lovely Professional University, Phagwara, under the guidance of Cherry Khosla, during August – December, 2023. All the information furnished in this project report is based on our own intensive work and is genuine.

Question Number:-08
Name of Student:-Sai Ganesh
Registration Number:-12222126

QUESTION:-

Student to implement a simulation program that simulates a computer's memory management system. The program should include a memory manager that uses a specific memory allocation algorithm (e.g. First-Fit, Best-Fit, or Worst-Fit) to allocate blocks of memory to processes. The students should also include a mechanism for deallocating memory when a process completes. The simulation should run for a set amount of time and record the average amount

of fragmentation and the number of wasted memory blocks at the end of each time unit. The below mentioned outcomes are expected:

- Memory allocation: The program should demonstrate the ability to allocate blocks of memory to processes using the chosen memory allocation algorithm (e.g. First-Fit, Best-Fit, or Worst-Fit)
- Memory deallocation: The program should include a mechanism for deallocating memory when a process completes, such as a "garbage collector" or a "free list.
- Fragmentation: The program should record the average amount of fragmentation at the end of each time unit. Fragmentation occurs when there are small, unused blocks of memory scattered throughout the memory space
- Wasted memory: The program should also record the number of wasted memory blocks at the end of each time unit. Wasted memory refers to blocks of memory that are no longer being used by any processes.
- Simulation results: The program should run the simulation for a set amount of time and display the results, including the average amount of fragmentation and the number of wasted memory blocks, at the end of each time unit. The students should also experiment with different input scenarios and algorithms to compare

INTRODUCTION :-

The memory management system simulation project aims to replicate the functionality of a computer's memory management system, allowing for the allocation and deallocation of memory blocks to various processes. This simulation serves as an educational tool, providing insight into the complexities of memory allocation algorithms and the challenges associated with managing memory in a computing environment.

The project demonstrates the implementation of a memory manager utilizing the First-Fit memory allocation algorithm. This algorithm efficiently allocates memory blocks to incoming processes based on their size requirements, ensuring optimal

utilization of available memory space. Moreover, the simulation tracks and reports metrics such as fragmentation and wasted memory, providing valuable insights into the efficiency and effectiveness of the memory management system.

By running the simulation for a specified period, the project aims to showcase how different input scenarios and memory allocation algorithms impact the overall performance of the memory manager. Through a systematic comparison of results, the project aims to highlight the importance of selecting appropriate memory allocation strategies to minimize fragmentation and optimize memory usage.

The development of this simulation project serves as a valuable learning experience for understanding fundamental concepts related to memory management in computer systems. It provides an opportunity for students to gain practical insights into memory allocation algorithms, fragmentation management, and the challenges associated with memory optimization in real-world computing scenarios. Furthermore, the project encourages experimentation and exploration, fostering a deeper understanding of the intricacies involved in memory management within a computing environment.

Methodology :-

1. Problem Identification:

Define the purpose and objectives of the project, identifying the key challenges and requirements associated with simulating a computer's memory management system.

2. Literature Review:

Conduct a comprehensive review of existing literature, research papers, and resources related to memory management, memory allocation algorithms, and

simulation methodologies. Gain insights into different approaches and best practices employed in similar projects.

3. System Design:

Define the structure and components of the memory management system simulation. Identify the data structures and algorithms required for memory allocation, deallocation, and tracking of fragmentation and wasted memory. Design the simulation framework that allows for the creation, allocation, and release of memory blocks.

4. Algorithm Implementation:

Implement the chosen memory allocation algorithm, such as First-Fit, Best-Fit, or Worst-Fit, within the simulation framework. Ensure the algorithm functions correctly and efficiently allocates memory blocks to incoming processes based on their size requirements.

5. Simulation Development:

Develop the simulation program that incorporates the memory management system, allowing for the allocation and deallocation of memory blocks. Create mechanisms to record fragmentation and wasted memory, and set up the simulation loop to run for a specified period, tracking the system's performance over time.

6. Testing and Debugging:

Conduct thorough testing to ensure the simulation program operates as intended. Validate the functionality of memory allocation, deallocation, and the accuracy of recorded metrics. Identify and resolve any bugs or errors within the simulation framework.

7. Performance Analysis:

Evaluate the performance of the memory management system by running the simulation with different input scenarios and memory allocation algorithms. Analyze and compare the recorded metrics to understand the impact of varying factors on system efficiency and memory utilization.

8. Documentation:

Document the entire development process, including the design, implementation, testing, and performance analysis of the simulation project. Provide clear and detailed explanations of the methodologies employed, the rationale behind design decisions, and the insights gained from the simulation results.

PSEUDO CODE :-

```
#include <stdio.h>
#include <stdbool.h>

#define TOTAL_MEMORY 1000

struct MemoryBlock {
   int start;
   int end;
   int process_id;
   bool free;
};

struct MemoryBlock memory_blocks[1000];
int block_count = 1;

void initialize_memory() {
```

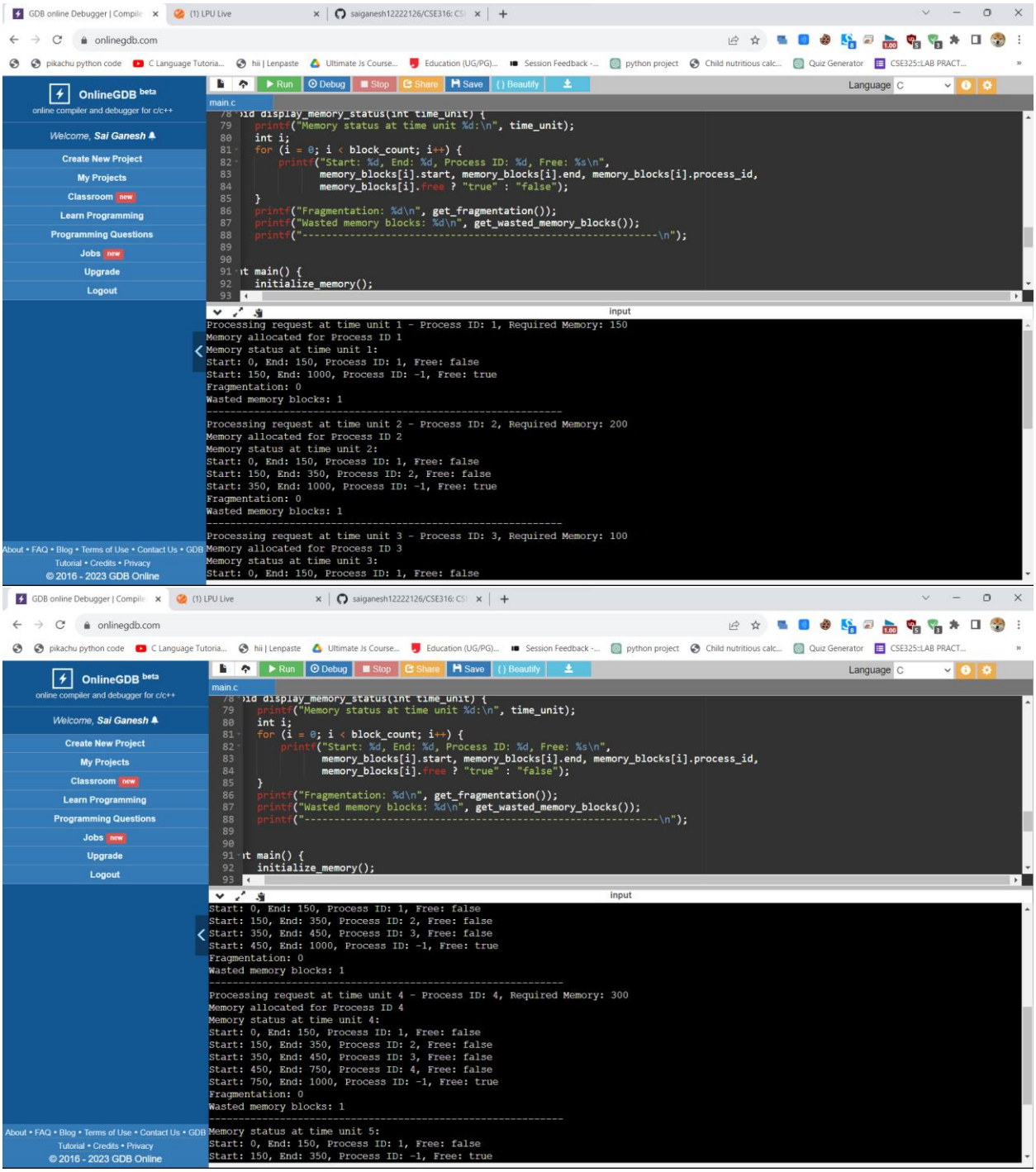
```
    memory_blocks[0].start = 0;
    memory_blocks[0].end = TOTAL_MEMORY;
    memory_blocks[0].process_id = -1;
    memory_blocks[0].free = true;
}

bool first_fit_allocation(int process_id, int required_memory) {
    int i, j;
    for (i = 0; i < block_count; i++) {
        if (memory_blocks[i].free && memory_blocks[i].end - memory_blocks[i].start >= required_memory) {
            memory_blocks[i].process_id = process_id;
            memory_blocks[i].free = false;
            if (memory_blocks[i].end - memory_blocks[i].start > required_memory) {
                for (j = block_count; j > i + 1; j--) {
                    memory_blocks[j] = memory_blocks[j - 1];
                }
                memory_blocks[i + 1].start = memory_blocks[i].start + required_memory;
                memory_blocks[i + 1].end = memory_blocks[i].end;
                memory_blocks[i].end = memory_blocks[i].start + required_memory;
                memory_blocks[i + 1].process_id = -1;
                memory_blocks[i + 1].free = true;
                block_count++;
            }
            return true;
        }
    }
    return false;
}

void deallocate_memory(int process_id) {
    int i;
    for (i = 0; i < block_count; i++) {
        if (memory_blocks[i].process_id == process_id) {
            memory_blocks[i].process_id = -1;
            memory_blocks[i].free = true;
```

```
        }
    }
}

int get_fragmentation() {
    int fragmentation = 0;
    int i;
    for (i = 0; i < block_count - 1; i++) {
        if (memory_blocks[i].free && memory_blocks[i + 1].free) {
            fragmentation += memory_blocks[i + 1].start - memory_blocks[i].end;
        }
    }
    return fragmentation;
}

int get_wasted_memory_blocks() {
    int wasted_blocks = 0;
    int i;
    for (i = 0; i < block_count; i++) {
        if (memory_blocks[i].free) {
            wasted_blocks++;
        }
    }
    return wasted_blocks;
}

void display_memory_status(int time_unit) {
    printf("Memory status at time unit %d:\n", time_unit);
    int i;
    for (i = 0; i < block_count; i++) {
        printf("Start: %d, End: %d, Process ID: %d, Free: %s\n",
               memory_blocks[i].start, memory_blocks[i].end, memory_blocks[i].process_id,
               memory_blocks[i].free ? "true" : "false");
    }
    printf("Fragmentation: %d\n", get_fragmentation());
```

```
    printf("Wasted memory blocks: %d\n", get_wasted_memory_blocks());
    printf("-----------------------------------------------------------\n");
}

int main() {
    initialize_memory();
    int processes[][2] = {{1, 150}, {2, 200}, {3, 100}, {4, 300}};
    int num_processes = sizeof(processes) / sizeof(processes[0]);

    for (int time_unit = 1; time_unit <= num_processes; time_unit++) {
        int process_id = processes[time_unit - 1][0];
        int required_memory = processes[time_unit - 1][1];

        printf("Processing request at time unit %d - Process ID: %d, Required Memory: %d\n", time_unit, process_id, required_memory);
        if (first_fit_allocation(process_id, required_memory)) {
            printf("Memory allocated for Process ID %d\n", process_id);
        } else {
            printf("Memory allocation failed for Process ID %d\n", process_id);
        }
        display_memory_status(time_unit);
    }

    // Deallocate memory for completed processes
    deallocate_memory(2);
    display_memory_status(num_processes + 1);

    return 0;
}
```

GITHUB LINK: https://github.com/saiganesh12222126/CSE316

Snap Shots of Output:



Conclusion:-

In conclusion, the development of the memory management system simulation project has provided valuable insights into the complexities of memory allocation and management within a computing environment. Through the implementation of the First-Fit memory allocation algorithm and the simulation of memory allocation and deallocation processes, this project has successfully showcased the challenges and considerations involved in optimizing memory usage and minimizing fragmentation.

The simulation results have highlighted the importance of selecting an appropriate memory allocation algorithm based on specific use cases and the size requirements of incoming processes. The comparison of fragmentation and wasted memory metrics has emphasized the significance of efficient memory management in maintaining system performance and ensuring optimal resource utilization.

Furthermore, the project has served as an educational tool, offering students a practical understanding of fundamental memory management concepts and algorithms. By engaging in the simulation development process and analyzing the simulation results, students have gained valuable experience in applying theoretical knowledge to real-world scenarios, thereby enhancing their comprehension of memory management strategies and their implications for system performance.

Moving forward, this project's insights and findings can be utilized to further explore advanced memory management techniques and algorithms, fostering continuous improvements in memory allocation efficiency and system optimization. Additionally, the project's documentation serves as a valuable resource for future reference and

provides a foundation for further research and development in the field of memory management systems.