B.BHANUTEJA REDDY-192325016

13. Construct a C program for implementation of the various memory allocation strategies.

Aim:

To implement various memory allocation strategies in C, including First-Fit, Best-Fit, and Worst-Fit, which are used for dynamic memory management in operating systems.

Algorithm:

1. **First-Fit**: Allocate memory to the first available block that is large enough.
2. **Best-Fit**: Allocate memory to the smallest block that can accommodate the request.
3. **Worst-Fit**: Allocate memory to the largest block available.
4. Each strategy will keep track of memory blocks, and when a request for memory is made, it will try to find the best suitable block using the strategy.
5. After allocation, the program should display the memory blocks, and when freeing memory, it should merge adjacent free blocks if necessary.

Procedure:

1. Define a structure for memory blocks.
2. Implement functions for First-Fit, Best-Fit, and Worst-Fit strategies.
3. Maintain a list of memory blocks with their status (allocated or free).
4. Allocate memory using the chosen strategy.
5. Free memory and merge adjacent free blocks.
6. Display memory allocation status after each operation.

CODE:

```
#include <stdio.h>

#include <stdlib.h>


#define MEMORY_SIZE 100
```

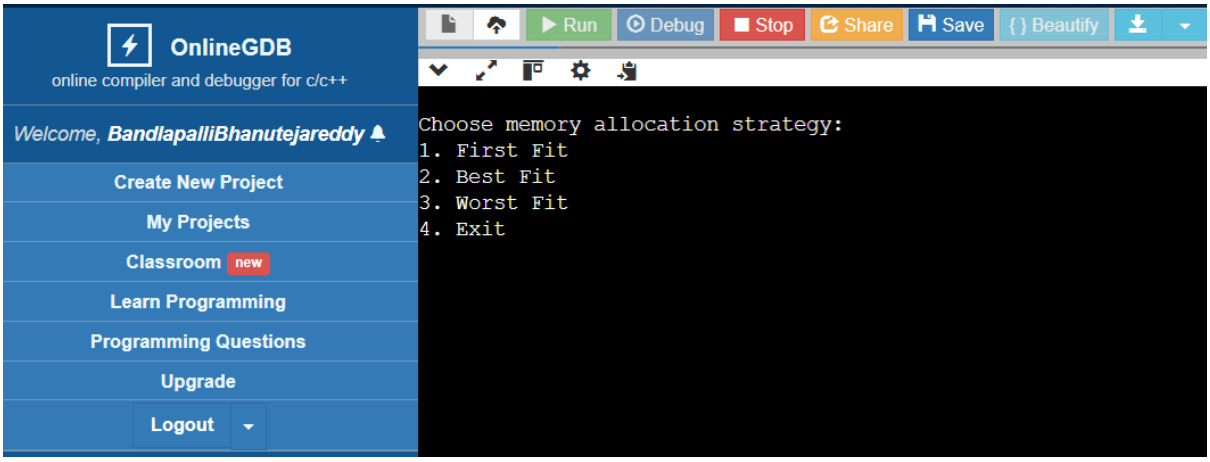
```c
typedef struct Block {
   int size;
   int is_allocated;
} Block;

Block memory[MEMORY_SIZE];

void initialize_memory() {
   for (int i = 0; i < MEMORY_SIZE; i++) {
      memory[i].size = 0;
      memory[i].is_allocated = 0;
   }
}

int first_fit(int size) {
   for (int i = 0; i < MEMORY_SIZE; i++) {
      if (!memory[i].is_allocated && memory[i].size >= size) {
         memory[i].is_allocated = 1;
         return i;
      }
   }
   return -1;
}

int best_fit(int size) {
   int best_idx = -1;
   for (int i = 0; i < MEMORY_SIZE; i++) {
      if (!memory[i].is_allocated && memory[i].size >= size) {
```

```
        if (best_idx == -1 || memory[i].size < memory[best_idx].size) {
          best_idx = i;
        }
      }
    }
    if (best_idx != -1) {
      memory[best_idx].is_allocated = 1;
    }
    return best_idx;
}

int worst_fit(int size) {
    int worst_idx = -1;
    for (int i = 0; i < MEMORY_SIZE; i++) {
      if (!memory[i].is_allocated && memory[i].size >= size) {
        if (worst_idx == -1 || memory[i].size > memory[worst_idx].size) {
          worst_idx = i;
        }
      }
    }
    if (worst_idx != -1) {
      memory[worst_idx].is_allocated = 1;
    }
    return worst_idx;
}

void free_block(int index) {
    memory[index].is_allocated = 0;
```

```
}

void display_memory() {
    printf("\nMemory Blocks: \n");
    for (int i = 0; i < MEMORY_SIZE; i++) {
        printf("Block %d: Size = %d, Allocated = %s\n", i, memory[i].size,
memory[i].is_allocated ? "Yes" : "No");
    }
}

int main() {
    initialize_memory();

    memory[0].size = 50;  // First block of size 50
    memory[1].size = 30;  // Second block of size 30
    memory[2].size = 70;  // Third block of size 70
    memory[3].size = 60;  // Fourth block of size 60

    int choice, size, block_index;

    while (1) {
        printf("\nChoose memory allocation strategy:\n");
        printf("1. First Fit\n2. Best Fit\n3. Worst Fit\n4. Exit\n");
        scanf("%d", &choice);

        if (choice == 4) break;

        printf("Enter the size of the block to allocate: ");
```

```c
        scanf("%d", &size);

        switch (choice) {
            case 1:
                block_index = first_fit(size);
                break;
            case 2:
                block_index = best_fit(size);
                break;
            case 3:
                block_index = worst_fit(size);
                break;
            default:
                printf("Invalid choice.\n");
                continue;
        }

        if (block_index != -1) {
            printf("Memory allocated at block %d\n", block_index);
        } else {
            printf("No suitable block found for the requested size.\n");
        }

        display_memory();
    }

    return 0;
}
```

OUTPUT:


OnlineGDB
online compiler and debugger for c/c++
Welcome, BandlapalliBhanutejareddy
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
Choose memory allocation strategy:
1. First Fit
2. Best Fit
3. Worst Fit
4. Exit