B. Siva Shirish-192324016

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 <limits.h>

void firstFit(int blockSize[], int m, int processSize[], int n) {
   int allocation[n];
   for (int i = 0; i < n; i++) {
      allocation[i] = -1; // No block allocated initially
   }
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    printf("\nFirst Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++) {
        allocation[i] = -1; // No block allocated initially
    }

    for (int i = 0; i < n; i++) {
        int bestIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx]) {
                    bestIdx = j;
                }
            }
        }
        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= processSize[i];
        }
    }

    printf("\nBest Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
```

```
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++) {
        allocation[i] = -1; // No block allocated initially
    }

    for (int i = 0; i < n; i++) {
        int worstIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx]) {
                    worstIdx = j;
                }
            }
        }
        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            blockSize[worstIdx] -= processSize[i];
        }
    }

    printf("\nWorst Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main() {
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
```

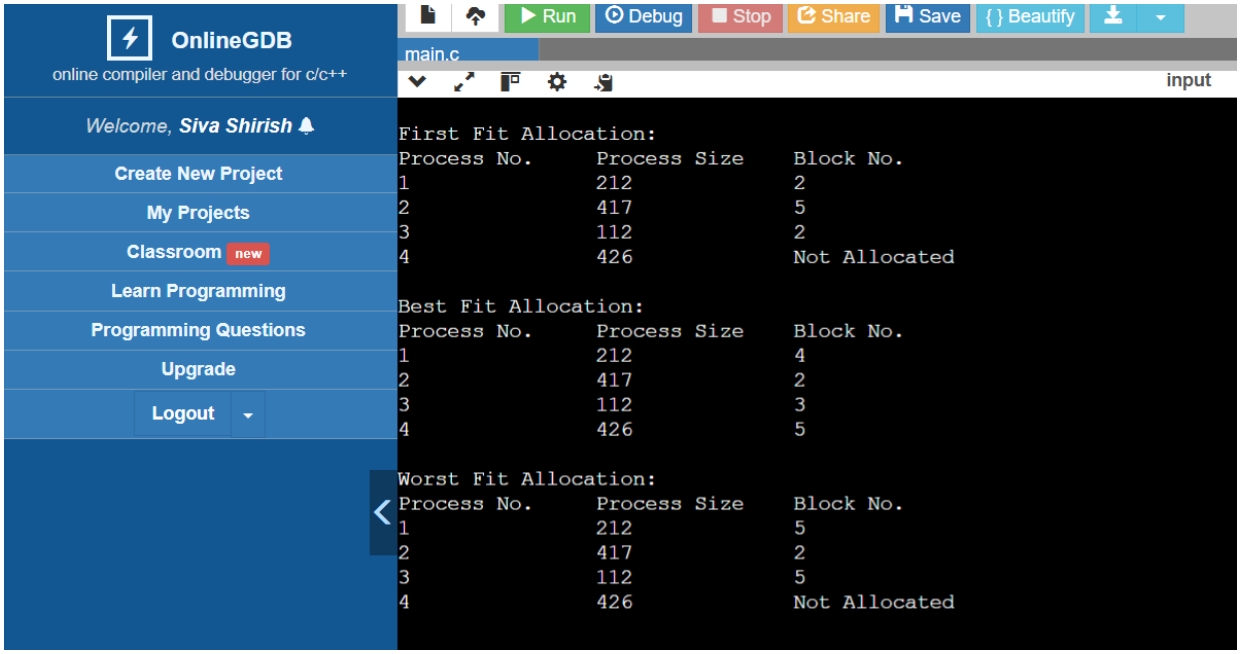
```
int m = sizeof(blockSize) / sizeof(blockSize[0]);
int n = sizeof(processSize) / sizeof(processSize[0]);

// Duplicate blockSize for each strategy
int blockSizeFirstFit[m], blockSizeBestFit[m], blockSizeWorstFit[m];
for (int i = 0; i < m; i++) {
   blockSizeFirstFit[i] = blockSize[i];
   blockSizeBestFit[i] = blockSize[i];
   blockSizeWorstFit[i] = blockSize[i];
}

firstFit(blockSizeFirstFit, m, processSize, n);
bestFit(blockSizeBestFit, m, processSize, n);
worstFit(blockSizeWorstFit, m, processSize, n);

return 0;
```

OUTPUT:


OnlineGDB
online compiler and debugger for c/c++
Welcome, Siva Shirish
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
main.c
input
First Fit Allocation:
Process No. Process Size Block No.
1 212 2
2 417 5
3 112 2
4 426 Not Allocated
Best Fit Allocation:
Process No. Process Size Block No.
1 212 4
2 417 2
3 112 3
4 426 5
Worst Fit Allocation:
Process No. Process Size Block No.
1 212 5
2 417 2
3 112 5
4 426 Not Allocated