Name : Bramha Nimbalkar

Roll no: 7

Div :E

ASSIGNMENT 9

Problem : Implement in C language First-fit, Next-fit, Best fit and worst fit algorithm for contiguous memory allocation.

```c
#include <stdio.h>

#define MAX_MEM_SIZE 1000

typedef struct {
    int s;
    int sz;
    int a;
} MBlock;

void initMem(MBlock mem[], int size);
void displayMem(MBlock mem[], int size);
int firstFit(MBlock mem[], int size, int pSize);
int nextFit(MBlock mem[], int size, int pSize, int *lastIdx);
int bestFit(MBlock mem[], int size, int pSize);
int worstFit(MBlock mem[], int size, int pSize);

int main() {
    MBlock mem[MAX_MEM_SIZE];
    initMem(mem, MAX_MEM_SIZE);

    int pSizes[] = {200, 100, 800, 50, 150, 1000};

    printf("First Fit Algorithm:\n");
    for (int i = 0; i < sizeof(pSizes) / sizeof(pSizes[0]); i++) {
        int idx = firstFit(mem, MAX_MEM_SIZE, pSizes[i]);
        if (idx != -1) {
            printf("Allocated: %d at position %d\n", pSizes[i],
mem[idx].s);
        } else {
            printf("Failed to allocate: %d\n", pSizes[i]);
        }
    }
```

```
    initMem(mem, MAX_MEM_SIZE);

    printf("\nNext Fit Algorithm:\n");
    int lastIdx = 0;
    for (int i = 0; i < sizeof(pSizes) / sizeof(pSizes[0]); i++) {
        int idx = nextFit(mem, MAX_MEM_SIZE, pSizes[i], &lastIdx);
        if (idx != -1) {
            printf("Allocated: %d at position %d\n", pSizes[i],
mem[idx].s);
        } else {
            printf("Failed to allocate: %d\n", pSizes[i]);
        }
    }

    initMem(mem, MAX_MEM_SIZE);

    printf("\nBest Fit Algorithm:\n");
    for (int i = 0; i < sizeof(pSizes) / sizeof(pSizes[0]); i++) {
        int idx = bestFit(mem, MAX_MEM_SIZE, pSizes[i]);
        if (idx != -1) {
            printf("Allocated: %d at position %d\n", pSizes[i],
mem[idx].s);
        } else {
            printf("Failed to allocate: %d\n", pSizes[i]);
        }
    }

    initMem(mem, MAX_MEM_SIZE);

    printf("\nWorst Fit Algorithm:\n");
    for (int i = 0; i < sizeof(pSizes) / sizeof(pSizes[0]); i++) {
        int idx = worstFit(mem, MAX_MEM_SIZE, pSizes[i]);
        if (idx != -1) {
            printf("Allocated: %d at position %d\n", pSizes[i],
mem[idx].s);
        } else {
            printf("Failed to allocate: %d\n", pSizes[i]);
        }
    }

    return 0;
}

void initMem(MBlock mem[], int size) {
    for (int i = 0; i < size; i++) {
        mem[i].s = i;
        mem[i].sz = MAX_MEM_SIZE - i;
        mem[i].a = 0;
    }
}
```

```
void displayMem(MBlock mem[], int size) {
    for (int i = 0; i < size; i++) {
        printf("Block %d: Start = %d, Size = %d, Allocated = %d\n", i,
mem[i].s, mem[i].sz, mem[i].a);
    }
}

int firstFit(MBlock mem[], int size, int pSize) {
    for (int i = 0; i < size; i++) {
        if (!mem[i].a && mem[i].sz >= pSize) {
            mem[i].sz = pSize;
            mem[i].a = 1;
            return i;
        }
    }
    return -1;
}

int nextFit(MBlock mem[], int size, int pSize, int *lastIdx) {
    for (int i = *lastIdx; i < size; i++) {
        if (!mem[i].a && mem[i].sz >= pSize) {
            mem[i].sz = pSize;
            mem[i].a = 1;
            *lastIdx = i;
            return i;
        }
    }

    for (int i = 0; i < *lastIdx; i++) {
        if (!mem[i].a && mem[i].sz >= pSize) {
            mem[i].sz = pSize;
            mem[i].a = 1;
            *lastIdx = i;
            return i;
        }
    }
    return -1;
}

int bestFit(MBlock mem[], int size, int pSize) {
    int bestIdx = -1;
    int bestSz = MAX_MEM_SIZE + 1;

    for (int i = 0; i < size; i++) {
        if (!mem[i].a && mem[i].sz >= pSize) {
            if (mem[i].sz < bestSz) {
                bestSz = mem[i].sz;
                bestIdx = i;
            }
        }
    }
```

```
    if (bestIdx != -1) {
        mem[bestIdx].sz = pSize;
        mem[bestIdx].a = 1;
        return bestIdx;
    }

    return -1;
}

int worstFit(MBlock mem[], int size, int pSize) {
    int worstIdx = -1;
    int worstSz = -1;

    for (int i = 0; i < size; i++) {
        if (!mem[i].a && mem[i].sz >= pSize) {
            if (mem[i].sz > worstSz) {
                worstSz = mem[i].sz;
                worstIdx = i;
            }
        }
    }

    if (worstIdx != -1) {
        mem[worstIdx].sz = pSize;
        mem[worstIdx].a = 1;
        return worstIdx;
    }

    return -1;
}
```

Output

```
[Running] cd "c:\Users\Lenovo\Desktop\" && gcc
First Fit Algorithm:
Allocated: 200 at position 0
Allocated: 100 at position 1
Allocated: 800 at position 2
Allocated: 50 at position 3
Allocated: 150 at position 4
Failed to allocate: 1000

Next Fit Algorithm:
Allocated: 200 at position 0
Allocated: 100 at position 1
Allocated: 800 at position 2
Allocated: 50 at position 3
Allocated: 150 at position 4
Failed to allocate: 1000

Best Fit Algorithm:
Allocated: 200 at position 800
Allocated: 100 at position 900
Allocated: 800 at position 200
Allocated: 50 at position 950
Allocated: 150 at position 850
Allocated: 1000 at position 0

Worst Fit Algorithm:
Allocated: 200 at position 0
Allocated: 100 at position 1
Allocated: 800 at position 2
Allocated: 50 at position 3
Allocated: 150 at position 4
Failed to allocate: 1000

[Done] exited with code=0 in 1.493 seconds
```