**PHẦN 3: CẤP PHÁT VÙNG NHỚ**

Tạo tệp baitap.c:

$ gedit baitap.c

Chèn đoạn code

#include

// Hàm sao chép mảng A sang mảng B

void makeCopyArray(int A[], int B[], int size) {

int i;

for (i = 0; i < size; i++) {

B[i] = A[i];

}

}

// Thuật toán First Fit

void firstFit(int partitionInput[], int m, int processInput[], int n) {

int i, j;

int partition[m], process[n];

// Sao chép mảng đầu vào vào mảng tạm thời

makeCopyArray(partitionInput, partition, m);

makeCopyArray(processInput, process, n);

int allocation[n];

for (i = 0; i < n; i++) {

allocation[i] = -1; }

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

if (partition[j] >= process[i]) {

allocation[i] = j; partition[j] -= process[i]; break;

}

}

}

// In ra kết quả

printf("\nProcess No.\tProcess Size\t\t\tPartition no.\n");

for (int i = 0; i < n; i++) {

printf(" %i\t\t\t", i + 1);

printf("%i\t\t\t\t", process[i]);

if (allocation[i] != -1)

printf("%i", allocation[i] + 1);

else

printf("Not Allocated");

printf("\n");

}

}

// Thuật toán Best Fit

void bestFit(int partitionInput[], int m, int processInput[], int n) {

int i, j; int partition[m], process[n];

// Sao chép mảng đầu vào vào mảng tạm thời

makeCopyArray(partitionInput, partition, m);

makeCopyArray(processInput, process, n);

int allocation[n];

for (i = 0; i < n; i++) {

allocation[i] = -1; }

for (i = 0; i < n; i++) {

int bestIndex = -1;

for (j = 0; j < m; j++) {

if (partition[j] >= process[i]) {

if (bestIndex == -1)

bestIndex = j;

else

if (partition[bestIndex] > partition[j])

bestIndex = j;

}

}

if (bestIndex != -1) {

allocation[i] = bestIndex;

partition[bestIndex] -= process[i];

}

}

// In ra kết quả

printf("\nProcess No.\tProcess Size\t\t\tPartition no.\n");

for (int i = 0; i < n; i++) {

printf(" %i\t\t\t", i + 1);

printf("%i\t\t\t\t", process[i]);

if (allocation[i] != -1)

printf("%i", allocation[i] + 1);

else

printf("Not Allocated");

printf("\n");

}

}

// Thuật toán Worst Fit

void worstFit(int partitionInput[], int m, int processInput[], int n) {

int i, j; int partition[m], process[n];

// Sao chép mảng đầu vào vào mảng tạm thời

makeCopyArray(partitionInput, partition, m);

makeCopyArray(processInput, process, n);

int allocation[n];

for (i = 0; i < n; i++) {

allocation[i] = -1;

}

for (i = 0; i < n; i++) {

int worstIndex = -1;

for (j = 0; j < m; j++) {

if (partition[j] >= process[i]) {

if (worstIndex == -1) worstIndex = j;

else

if (partition[worstIndex] < partition[j]) worstIndex = j;

}

}

if (worstIndex != -1) {

allocation[i] = worstIndex;

partition[worstIndex] -= process[i];

}

}

// In ra kết quả

printf("\nProcess No.\tProcess Size\t\t\tPartition no.\n");

for (int i = 0; i < n; i++) {

printf(" %i\t\t\t", i + 1);

printf("%i\t\t\t\t", process[i]);

if (allocation[i] != -1)

printf("%i", allocation[i] + 1);

else

printf("Not Allocated"); printf("\n");

}

}

int main() {

int m, n;

int i;

printf("Enter number of partitions: ");

scanf("%d", &n);

int partition[n];

printf("Enter list of partition sizes: ");

for (i = 0; i < n; i++) {

scanf("%d", &partition[i]);

}

printf("Enter number of processes: ");

scanf("%d", &m);

int process[m];

printf("Enter list of process sizes: ");

for (i = 0; i < m; i++) {

scanf("%d", &process[i]);

}

printf("\n\n\t\t---- FIRST FIT ALGORITHM ----\n");

firstFit(partition, n, process, m);

printf("\n\n\t\t---- BEST FIT ALGORITHM ----\n");

bestFit(partition, n, process, m);

printf("\n\n\t\t---- WORST FIT ALGORITHM ----\n");

worstFit(partition, n, process, m);

return 0;

}