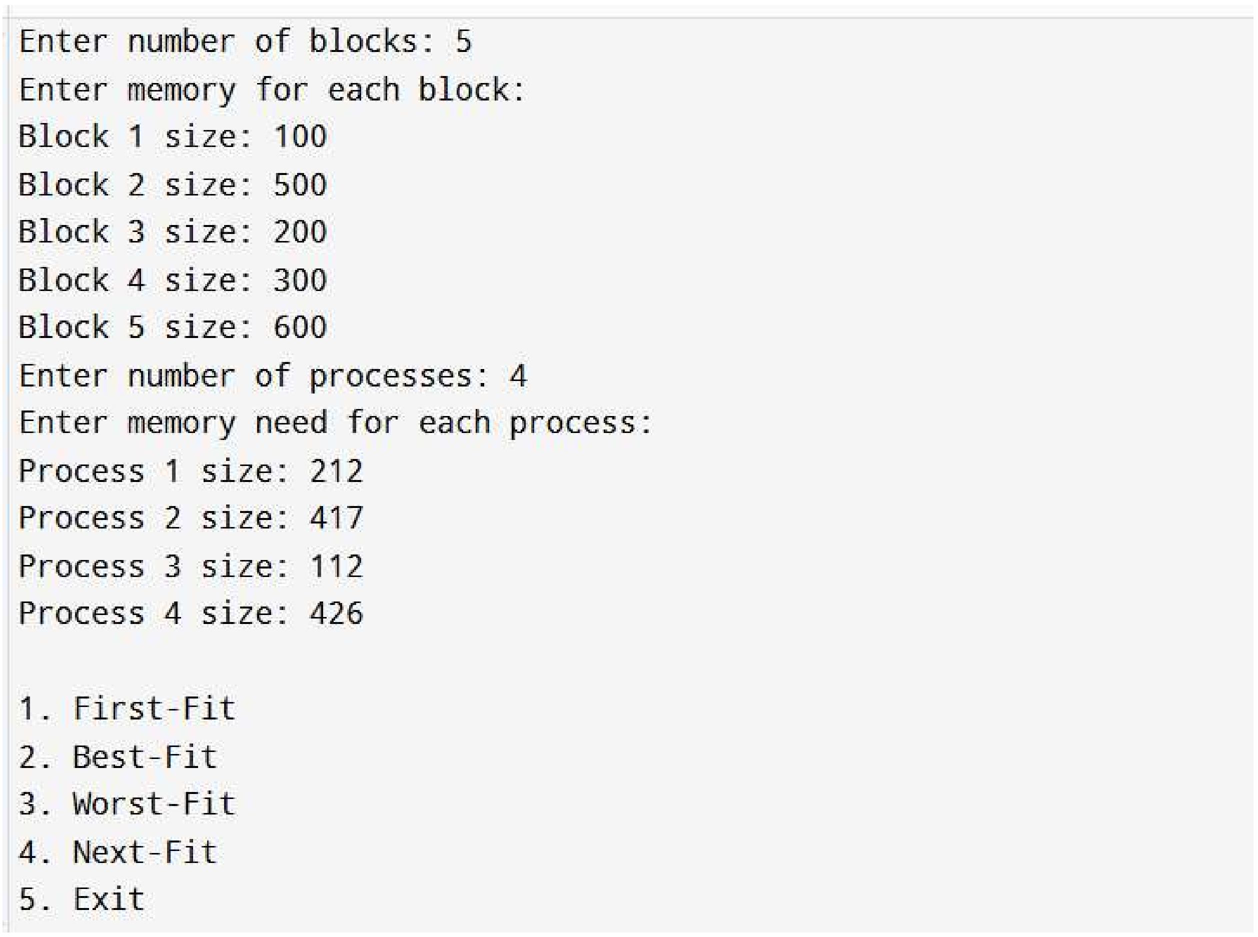
ASSIGNMENT-9

Write a program to allocate memory blocks to processes using the First Fit, Best Fit, Worst Fit, and Next-fit algorithms. The program should allocate memory blocks to each process according to the selected algorithm, display the allocation results, and compute both internal and external fragmentation.

```
#include<iostream>
using namespace std;

struct pro {
   int pid;
   int pmemo;
};

struct block {
   int bid;
   int memo;
};

void first(const pro p[], const block b[], int n, int x) {
   cout << "\n=== FIRST FIT ALLOCATION ===\n";


   block temp[n];
   for (int i = 0; i < n; i++) temp[i] = b[i];

   for (int i = 0; i < x; i++) {
      bool allocated = false;
      for (int j = 0; j < n; j++) {
         if (temp[j].memo >= p[i].pmemo) {
            cout << "Process " << p[i].pid
                << " (size " << p[i].pmemo << ") -> Block "
                << temp[j].bid << " (remaining "
                << temp[j].memo - p[i].pmemo << ")\n";
            temp[j].memo -= p[i].pmemo; // change only in temp
            allocated = true;
```

```
                break;
            }
        }
        if (!allocated) {
            cout << "Process " << p[i].pid << " cannot be allocated.\n";
        }
    }
}

void best(const pro p[], const block b[], int n, int x) {
    cout << "\n=== BEST FIT ALLOCATION ===\n";


    block temp[n];
    for (int i = 0; i < n; i++) temp[i] = b[i];


    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (temp[j].memo < temp[i].memo) {
                swap(temp[i], temp[j]);
            }
        }
    }

    for (int i = 0; i < x; i++) {
        bool allocated = false;
        for (int j = 0; j < n; j++) {
            if (temp[j].memo >= p[i].pmemo) {
                cout << "Process " << p[i].pid
                     << " (size " << p[i].pmemo << ") -> Block "
                     << temp[j].bid << " (remaining "
                     << temp[j].memo - p[i].pmemo << ")\n";
                temp[j].memo -= p[i].pmemo;
                allocated = true;
                break;
            }
        }
        if (!allocated) {
            cout << "Process " << p[i].pid << " cannot be allocated.\n";
        }
    }
}

void worst(const pro p[], block b[], int n, int x) {
    cout << "\n=== WORST FIT ALLOCATION ===\n";

    // For each process
    for (int i = 0; i < x; i++) {
        int worstIdx = -1;


        for (int j = 0; j < n; j++) {
            if (b[j].memo >= p[i].pmemo) {
                if (worstIdx == -1 || b[j].memo > b[worstIdx].memo)
                    worstIdx = j;
            }
        }


        if (worstIdx != -1) {
            cout << "Process " << p[i].pid
                 << " (size " << p[i].pmemo << ") -> Block "
                 << b[worstIdx].bid << " (remaining "
                 << b[worstIdx].memo - p[i].pmemo << ")\n";

            b[worstIdx].memo -= p[i].pmemo;  // update remaining memory
        } else {
            cout << "Process " << p[i].pid << " cannot be allocated.\n";
        }
    }
}

void next(const pro p[], block b[], int n, int x) {
    cout << "\n=== NEXT FIT ALLOCATION ===\n";
    int lastAllocated = 0;

    for (int i = 0; i < x; i++) {
        bool allocated = false;
        int count = 0;


        for (int j = lastAllocated; count < n; j = (j + 1) % n, count++) {
            if (b[j].memo >= p[i].pmemo) {
                cout << "Process " << p[i].pid
                     << " (size " << p[i].pmemo << ") -> Block "
                     << b[j].bid << " (remaining "
                     << b[j].memo - p[i].pmemo << ")\n";
                b[j].memo -= p[i].pmemo;
                lastAllocated = j;
                allocated = true;
                break;
            }
        }

        if (!allocated) {
            cout << "Process " << p[i].pid << " cannot be allocated.\n";
        }
    }
}

int main() {
    int n;
    cout << "Enter number of blocks: ";
    cin >> n;
    block b[n];

    cout << "Enter memory for each block:\n";
    for (int i = 0; i < n; i++) {
        b[i].bid = i + 1;
        cout << "Block " << b[i].bid << " size: ";
    cin >> b[i].memo;
}

int x;
cout << "Enter number of processes: ";
cin >> x;
pro p[x];

cout << "Enter memory need for each process:\n";
for (int i = 0; i < x; i++) {
    p[i].pid = i + 1;
    cout << "Process " << p[i].pid << " size: ";
    cin >> p[i].pmemo;
}

int choice;
bool running = true;

while (running) {
    cout << "\n1. First-Fit\n2. Best-Fit\n3. Worst-Fit\n4. Next-Fit\n5. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            first(p, b, n, x);
            break;
        case 2:
            best(p, b, n, x);
            break;
        case 3:
            worst(p,b,n ,x);
            cout << "Worst-Fit not implemented yet.\n";
            break;
        case 4:
        next(p,b,n, x);
            cout << "Next-Fit not implemented yet.\n";
            break;
        case 5:
            running = false;
            break;
        default:
            cout << "Invalid choice!\n";
        }
    }

    return 0;
}
```

OUTPUT OF PROGRAM:



```
Enter number of blocks: 5
Enter memory for each block:
Block 1 size: 100
Block 2 size: 500
Block 3 size: 200
Block 4 size: 300
Block 5 size: 600
Enter number of processes: 4
Enter memory need for each process:
Process 1 size: 212
Process 2 size: 417
Process 3 size: 112
Process 4 size: 426

1. First-Fit
2. Best-Fit
3. Worst-Fit
4. Next-Fit
5. Exit
```

```
Enter your choice: 1

=== FIRST FIT ALLOCATION ===
Process 1 (size 212) -> Block 2 (remaining 76)
Process 2 cannot be allocated.
Process 3 (size 112) -> Block 3 (remaining 88)
Process 4 cannot be allocated.
```

```
Enter your choice: 2

=== BEST FIT ALLOCATION ===
Process 1 (size 212) -> Block 2 (remaining 76)
Process 2 cannot be allocated.
Process 3 (size 112) -> Block 3 (remaining 88)
Process 4 cannot be allocated.
```

```
Enter your choice: 3

=== WORST FIT ALLOCATION ===
Process 1 (size 212) -> Block 4 (remaining 88)
Process 2 cannot be allocated.
Process 3 (size 112) -> Block 2 (remaining 176)
Process 4 cannot be allocated.
Worst-Fit not implemented yet.
```

```
Enter your choice: 4

=== NEXT FIT ALLOCATION ===
Process 1 (size 212) -> Block 2 (remaining 288)
Process 2 (size 417) -> Block 5 (remaining 183)
Process 3 (size 112) -> Block 5 (remaining 71)
Process 4 cannot be allocated.
Next-Fit not implemented yet.
```