

Name: -Ashish Kumar

Reg no: -12115852 Section: -K21PP Roll no: -26

* **1.Problem:**

Student to implement a simulation program that simulates a computer's memory management system. The program should include a memory manager that uses a specific memory allocation algorithm (e.g., First-Fit, Best-Fit, or Worst-Fit) to allocate blocks of memory to processes. The students should also include a mechanism for deallocating memory when a process completes. The simulation should run for a set amount of time and record the average amount of fragmentation and the number of wasted memory blocks at the end of each time unit.

Answer: As I have to allocate the process in the memory manger in, so that each process should be allocated to their required memory space. And follow the memory allocation algorithm like first-fit, best-fit, worst-fit. I have used the C++ programming language to solve the program and allocate the memory according to the rules of algorithm.

**APPROACH**

* **FIRST\_FIT**
* I have used the First Fit memory allocation algorithm, that is a simple algorithm which allocates memory to a process using the first available memory block that can accommodate the current running process .
* First, I have declared the three arrays :- 1.block\_no , 2.process\_size and 3.block\_size.

**1.block\_no** store the block number of the each memory block.

**2.process\_size** it store the size of the each of the process.

**3.block\_size** it will stores the size of the each of the memory block.

* The program used the FOR loop which interates over each of the process and searches the first available block which can accomodat the current running process
* Each time procees iterates and searches block , if it not found any block which can’t accommodate the process of size then it prints that the process is not allocated.
* It also counts the number of the blocks are allocated to the process and the fragmented space .
* In this every time when the process is allocated to the block then the fragmented space is calculated and displayed over the result.
* The program also calculated the number of the wasted memory blocks and the average fragmented space .

**// here it will to keep track of fragmentation, allocated blocks, and unused blocks.**

**AlGORITHM FOR THE FIRST-FIT AND FOR THE AVG FRAGMENTATION AND THE NO OF**

**WASTED BLOCK:-**

int fragmentation = 0;

int allocated\_blocks = 0;

int unused\_blocks = 0;

// here I have allocated the process using the First Fit algorithm.

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

{

int temp=-1;

for (int j=0; j<n+1; j++)

{

if (process\_size[i]<=block\_size[j])

{

temp=j;

fragmentation += block\_size[j] - process\_size[i]; // add the fragmented space to total fragmentation

allocated\_blocks++; // here it will increment the counter for the allocated blocks

block\_size[j] -= process\_size[i]; // update the block size

cout<<"P"<<i+1<<"\t\t"<<process\_size[i]<<"\t\t\t"<<temp+1<<"\t\t"<<block\_size[j]<<endl;

break;

}

}

if(temp==-1)

{

cout<<"P"<<i+1<<"\t\t"<<process\_size[i]<<"\t\t\tNot Allocated"<<endl;

}

}

// Count the number of unused blocks.

for (int i=0; i<n+1; i++)

{

if (block\_size[i] > 0)

{

unused\_blocks++;

}

}

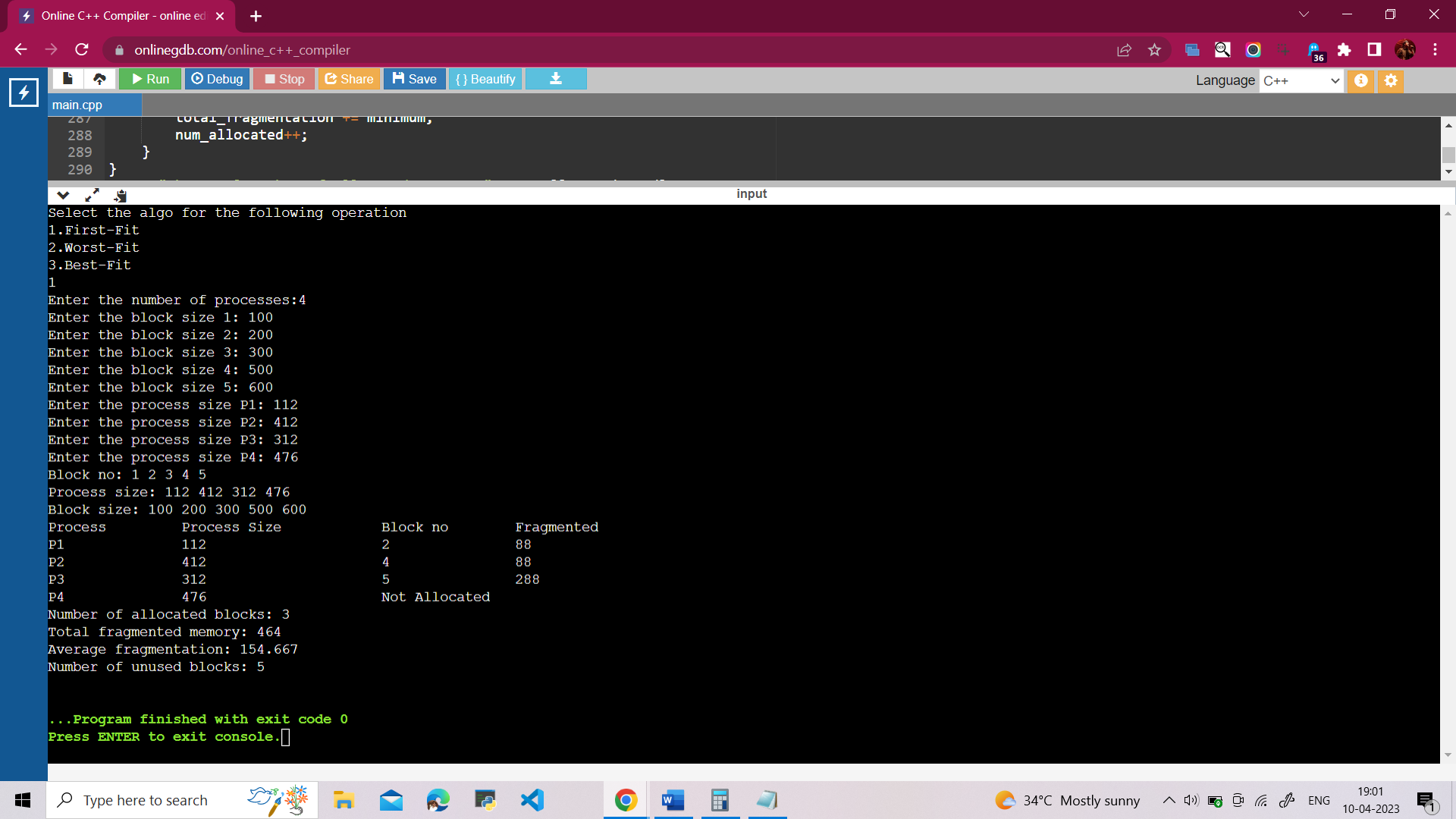
// Print out the total number of allocated blocks, the total fragmentation, and the number of unused blocks.

cout<<"Number of allocated blocks: "<<allocated\_blocks<<endl;

cout<<"Total fragmented memory: "<<fragmentation<<endl;

cout<<"Average fragmentation: "<<(double)fragmentation/allocated\_blocks<<endl;

cout<<"Number of unused blocks: "<<unused\_blocks<<endl**;**



* **WORST-FIT**
* In this we have used the worst-fir memory allocation algorithm.
* First, I have declared the three arrays :- 1.block\_no , 2.process\_size and 3.block\_size.

**1.block\_no** store the block number of the each memory block.

**2.process\_size** it store the size of the each of the process.

**3.block\_size** it will stores the size of the each of the memory block.

* In the algorithm each process iterates and searches for the block which can accommodate the process.
* In this each process searches the block with the largest size that will accommodate the process.
* If the process find the largest block with memory size then the process is allocated to that block and the size of the block is updated.
* If no any block found that can accommodate the process then that process is marked as the not allocated.
* In this every time when the process is allocated to the block then the fragmented space is calculated and displayed over the result.
* The program also calculated the number of the wasted memory blocks and the average fragmented space .

**AlGORITHM FOR THE WORST-FIT AND FOR THE AVG FRAGMENTATION AND THE NO OF**

**WASTED BLOCK:-**

**int total\_fragmented\_space = 0;**

**int num\_allocated\_processes = 0;**

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

**{**

**int max=0,temp=-1;**

**for(int j=0; j<n+1; j++)**

**{**

**if(block\_size[j]>process\_size[i])**

**{**

**int k=block\_size[j]-process\_size[i];**

**if(k>max)**

**{**

**max=k;**

**temp=j;**

**}**

**}**

**}**

**if(temp==-1)**

**{**

**cout<<"P"+to\_string(i+1)+"\t\t"+to\_string(process\_size[i])+"\t\t\tNot Allocated"<<endl;**

**fragmented\_space[i] = -1;**

**}**

**else**

**{**

**block\_size[temp] -= process\_size[i];**

**cout<<"P"+to\_string(i+1)+"\t\t"+to\_string(process\_size[i])+"\t\t\t"+to\_string(temp+1)+"\t\t\t"<<block\_size[temp]<<endl;**

**fragmented\_space[i] = block\_size[temp];**

**num\_allocated\_processes++;**

**total\_fragmented\_space += fragmented\_space[i];**

**}**

**}**

**cout << "Fragmented space: ";**

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

**{**

**if (fragmented\_space[i] == -1)**

**{**

**cout << "NA ";**

**}**

**else**

**{**

**cout << fragmented\_space[i] << " ";**

**}**

**}**

**cout << endl;**

**int wasted\_blocks = 0;**

**for (int i=0; i<n+1; i++)**

**{**

**if (block\_size[i] > 0)**

**{**

**wasted\_blocks++;**

**}**

**}**

**cout << "Number of wasted memory blocks: " << wasted\_blocks << endl;**

**if (num\_allocated\_processes > 0)**

**{**

**double avg\_fragmented\_space = static\_cast<double>(total\_fragmented\_space) / num\_allocated\_processes;**

**cout << "Average fragmented space: " << avg\_fragmented\_space << endl;**

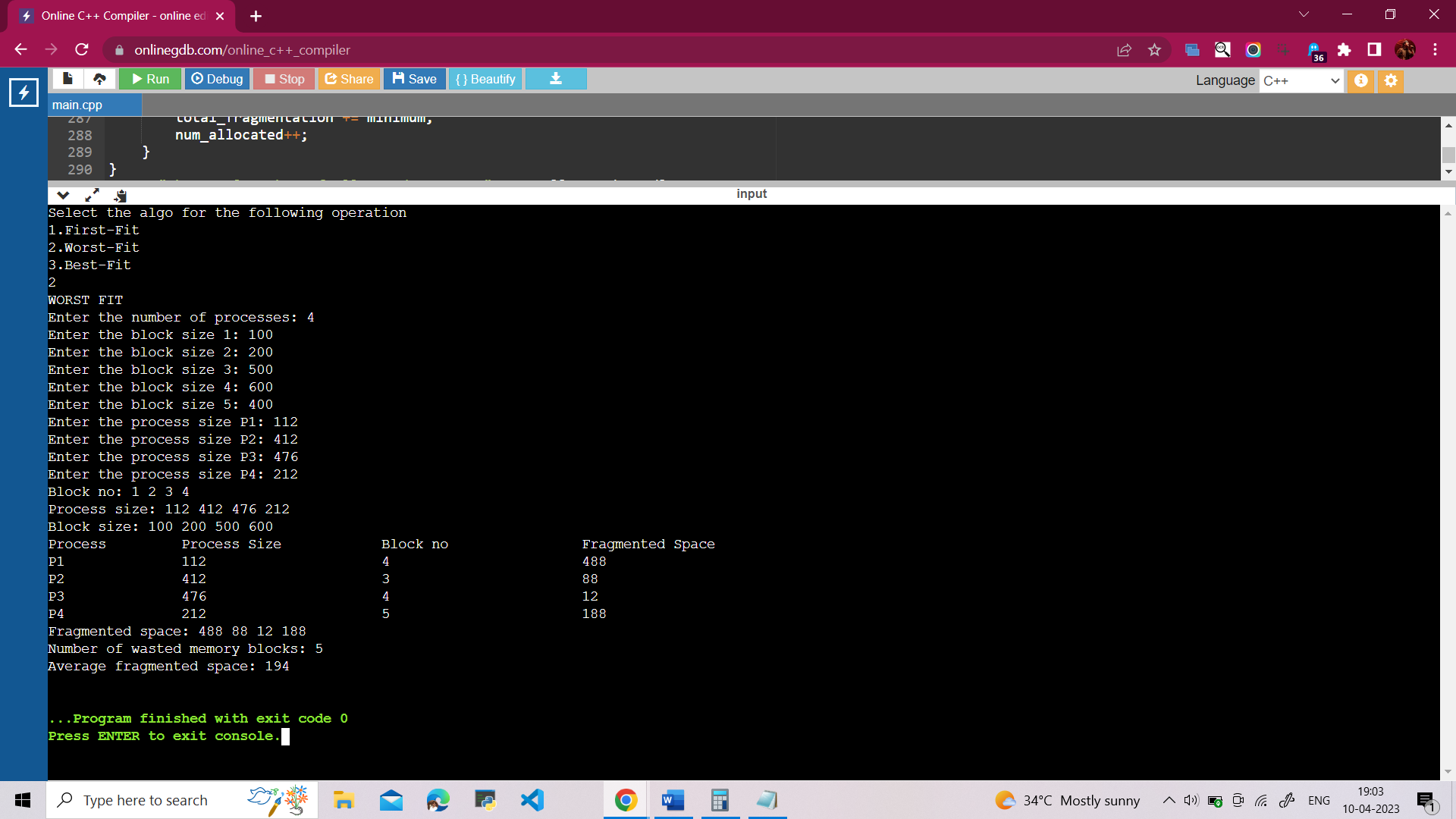
**}**

**else**

**{**

**cout << "No"<<endl;**

**}**



* **BEST\_FIT**

**Algorithm:**

* First, I have declared the three arrays :- 1.block\_no , 2.process\_size and 3.block\_size.

**1.block\_no** store the block number of the each memory block.

**2.process\_size** it store the size of the each of the process.

**3.block\_size** it will stores the size of the each of the memory block.

* After that I have asked the user to enter the number o the process they want to allocate.
* Next, user is asked to enter the size of the memory block, process sizes, and the block number.
* Then entered the size of each memory block using a for loop that will nuns n+1 times.
* After that I entered the size of each process again by using the for loop which will run for the n times.
* After the loop is completed the memory block, process sizes, and the block number are printed.
* For allocating the process to the memory I have used the loop using the Best Fit Algorithm.
* I have used the again one loop but the loop is inside another loop that will run till it will no find the smallest memory that fit the current running process.
* If the any block is found that will accommodate the process in that memory then the difference between the block size and the process size is calculated and the difference

stored in the variable minimum.

* The block that have minimum difference is selected for the memory allocation, and their size is set to 0 to that no new process tries to occupy.
* In this every time when the process is allocated to the block then the fragmented space is calculated and displayed over the result.
* The program also calculated the number of the wasted memory blocks and the average fragmented space .

**AlGORITHM FOR THE BEST-FIT AND FOR THE AVG FRAGMENTATION AND THE NO OF**

**WASTED BLOCK:-**

**int total\_fragmentation = 0;**

**int num\_allocated = 0;**

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

**{**

**int minimum = 9999999, temp;**

**for (int j=0; j<n+1; j++)**

**{**

**if (block\_size[j] > process\_size[i])**

**{**

**int k = block\_size[j] - process\_size[i];**

**if (abs(k) < minimum)**

**{**

**minimum = k;**

**temp = j;**

**}**

**}**

**}**

**if (minimum == 9999999)**

**{**

**cout << "P" << i+1 << "\t\t" << process\_size[i] << "\t\t\tNot Allocated" << endl;**

**}**

**else**

**{**

**block\_size[temp] -= process\_size[i];**

**cout << "P" << i+1 << "\t\t" << process\_size[i] << "\t\t\t" << temp+1 << "\t\t\t" << minimum << endl;**

**total\_fragmentation += minimum;**

**num\_allocated++;**

**}**

**}**

**cout<<"the total number of allocated process:"<<num\_allocated<<endl;**

**cout << "Average fragmentation: " << static\_cast<double>(total\_fragmentation)/num\_allocated << endl;**

**int unused\_blocks = 0;**

**for (int i=0; i<n+1; i++)**

**{**

**if (block\_size[i] > 0)**

**{**

**unused\_blocks++;**

**total\_fragmentation += block\_size[i];**

**}**

**}**

**cout << "Number of unused memory blocks: " << unused\_blocks << endl;**

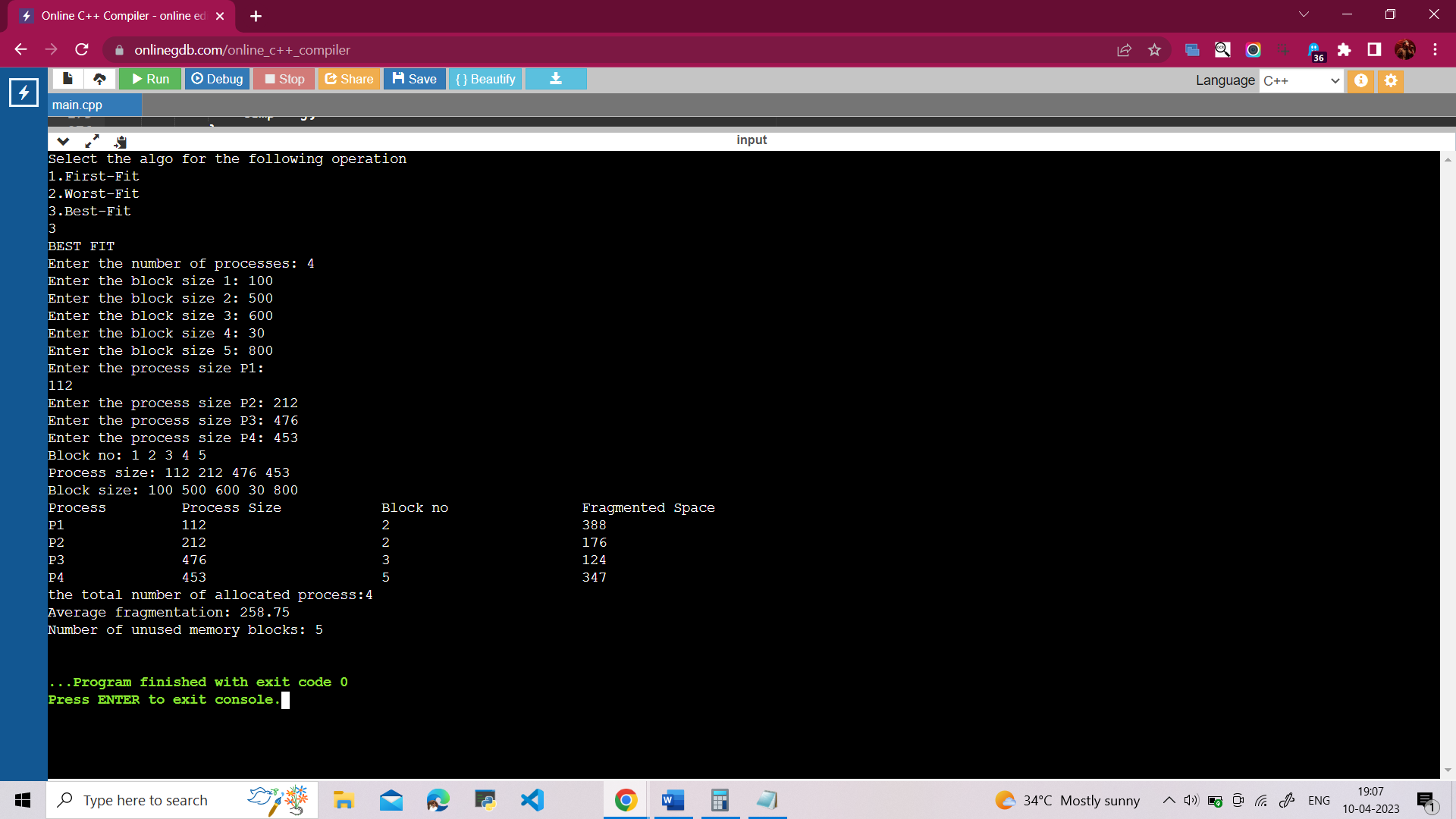
**}**

**}**

**}**

**}**

**PROGRAM EXECUTON :**



**PROGRAM EXECUTION CODE :**

#include<iostream>

using namespace std;

int main()

{

int select\_algo;

cout<<"Select the algo for the following operation"<<endl;

cout<<"1.First-Fit\n";

cout<<"2.Worst-Fit\n";

cout<<"3.Best-Fit\n";

cin>>select\_algo;

switch(select\_algo)

{

case 1:

{

Cout<<”FIRST-FIT”<<endl;

cout<<"Enter the number of processes:";

int n;

cin>>n;

// here declare the arrays to store block numbers, process sizes, and block sizes

int block\_no[n+1], process\_size[n], block\_size[n+1];

// here use the for loop to enter the block sizes.

for (int i=0;i<n+1;i++)

{

block\_no[i]=i+1;

cout<<"Enter the block size " << i+1 << ": ";

cin>>block\_size[i];

}

// here use the for loop to enter the process sizes.

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

{

cout<<"Enter the process size P" << i+1 << ": ";

cin>>process\_size[i];

}

// here to print the block numbers, process sizes, and block sizes entered.

cout<<"Block no: ";

for(int i=0;i<n+1; i++)

{

cout<<block\_no[i]<<" ";

}

cout<<endl;

cout<<"Process size: ";

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

{

cout<<process\_size[i]<<" ";

}

cout<<endl;

cout<<"Block size: ";

for(int i=0; i<n+1;i++)

{

cout<<block\_size[i]<<" ";

}

cout<<endl;

// Print the header for the allocation table.

cout<<"Process\t\tProcess Size\t\tBlock no\tFragmented"<<endl;

// Initialize variables to keep track of fragmentation, allocated blocks, and unused blocks.

int fragmentation = 0;

int allocated\_blocks = 0;

int unused\_blocks = 0;

// Allocate the processes using the First Fit algorithm.

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

{

int temp=-1;

for (int j=0; j<n+1; j++)

{

if (process\_size[i]<=block\_size[j])

{

temp=j;

fragmentation += block\_size[j] - process\_size[i]; // add the fragmented space to total fragmentation

allocated\_blocks++; // increment the counter for allocated blocks

block\_size[j] -= process\_size[i]; // update the block size

cout<<"P"<<i+1<<"\t\t"<<process\_size[i]<<"\t\t\t"<<temp+1<<"\t\t"<<block\_size[j]<<endl;

break;

}

}

if(temp==-1)

{

cout<<"P"<<i+1<<"\t\t"<<process\_size[i]<<"\t\t\tNot Allocated"<<endl;

}

}

// Count the number of unused blocks.

for (int i=0; i<n+1; i++)

{

if (block\_size[i] > 0)

{

unused\_blocks++;

}

}

// Print out the total number of allocated blocks, the total fragmentation, and the number of unused blocks.

cout<<"Number of allocated blocks: "<<allocated\_blocks<<endl;

cout<<"Total fragmented memory: "<<fragmentation<<endl;

cout<<"Average fragmentation: "<<(double)fragmentation/allocated\_blocks<<endl;

cout<<"Number of unused blocks: "<<unused\_blocks<<endl;

break;

}

{

case 2:

{

#include <iostream>

#include <string>

using namespace std;

int main() {

cout << "WORST FIT" << endl;

cout << "Enter the number of processes: ";

int n;

cin >> n;

int block\_no[n+1], process\_size[n], block\_size[n+1], fragmented\_space[n];

for (int i=0; i<n+1; i++)

{

block\_no[i] = i+1;

cout << "Enter the block size " << i+1 << ": ";

cin >> block\_size[i];

}

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

{

cout << "Enter the process size P" << i+1 << ": ";

cin >> process\_size[i];

}

cout << "Block no: ";

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

{

cout << block\_no[i] << " ";

}

cout << endl;

cout << "Process size: ";

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

{

cout << process\_size[i] << " ";

}

cout << endl;

cout << "Block size: ";

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

{

cout << block\_size[i] << " ";

}

cout << endl;

cout << "Process\t\tProcess Size\t\tBlock no\t\tFragmented Space" << endl;

int total\_fragmented\_space = 0;

int num\_allocated\_processes = 0;

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

{

int max=0,temp=-1;

for(int j=0; j<n+1; j++)

{

if(block\_size[j]>process\_size[i])

{

int k=block\_size[j]-process\_size[i];

if(k>max)

{

max=k;

temp=j;

}

}

}

if(temp==-1)

{

cout<<"P"+to\_string(i+1)+"\t\t"+to\_string(process\_size[i])+"\t\t\tNot Allocated"<<endl;

fragmented\_space[i] = -1;

}

else

{

block\_size[temp] -= process\_size[i];

cout<<"P"+to\_string(i+1)+"\t\t"+to\_string(process\_size[i])+"\t\t\t"+to\_string(temp+1)+"\t\t\t"<<block\_size[temp]<<endl;

fragmented\_space[i] = block\_size[temp];

num\_allocated\_processes++;

total\_fragmented\_space += fragmented\_space[i];

}

}

cout << "Fragmented space: ";

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

{

if (fragmented\_space[i] == -1)

{

cout << "NA ";

}

else

{

cout << fragmented\_space[i] << " ";

}

}

cout << endl;

int wasted\_blocks = 0;

for (int i=0; i<n+1; i++)

{

if (block\_size[i] > 0)

{

wasted\_blocks++;

}

}

cout << "Number of wasted memory blocks: " << wasted\_blocks << endl;

if (num\_allocated\_processes > 0)

{

double avg\_fragmented\_space = static\_cast<double>(total\_fragmented\_space) / num\_allocated\_processes;

cout << "Average fragmented space: " << avg\_fragmented\_space << endl;

}

else

{

cout << "No"<<endl;

}

}

}

break;

}

}

case 3:

{

cout << "BEST FIT" << endl;

cout << "Enter the number of processes: ";

int n;

cin >> n;

int block\_no[n+1], process\_size[n], block\_size[n+1];

for (int i=0; i<n+1; i++)

{

block\_no[i] = i+1;

cout << "Enter the block size " << i+1 << ": ";

cin >> block\_size[i];

}

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

{

cout << "Enter the process size P" << i+1 << ": ";

cin >> process\_size[i];

}

cout << "Block no: ";

for (int i=0; i<n+1; i++)

{

cout << block\_no[i] << " ";

}

cout << endl;

cout << "Process size: ";

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

{

cout << process\_size[i] << " ";

}

cout << endl;

cout << "Block size: ";

for (int i=0; i<n+1; i++)

{

cout << block\_size[i] << " ";

}

cout << endl;

cout << "Process\t\tProcess Size\t\tBlock no\t\tFragmented Space" << endl;

int total\_fragmentation = 0;

int num\_allocated = 0;

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

{

int minimum = 9999999, temp;

for (int j=0; j<n+1; j++)

{

if (block\_size[j] > process\_size[i])

{

int k = block\_size[j] - process\_size[i];

if (abs(k) < minimum)

{

minimum = k;

temp = j;

}

}

}

if (minimum == 9999999)

{

cout << "P" << i+1 << "\t\t" << process\_size[i] << "\t\t\tNot Allocated" << endl;

}

else

{

block\_size[temp] -= process\_size[i];

cout << "P" << i+1 << "\t\t" << process\_size[i] << "\t\t\t" << temp+1 << "\t\t\t" << minimum << endl;

total\_fragmentation += minimum;

num\_allocated++;

}

}

cout<<"the total number of allocated process:"<<num\_allocated<<endl;

cout << "Average fragmentation: " << static\_cast<double>(total\_fragmentation)/num\_allocated << endl;

int unused\_blocks = 0;

for (int i=0; i<n+1; i++)

{

if (block\_size[i] > 0)

{

unused\_blocks++;

total\_fragmentation += block\_size[i];

}

}

cout << "Number of unused memory blocks: " << unused\_blocks << endl;

cout << "Unused memory: " << total\_fragmentation << endl;}

}

}