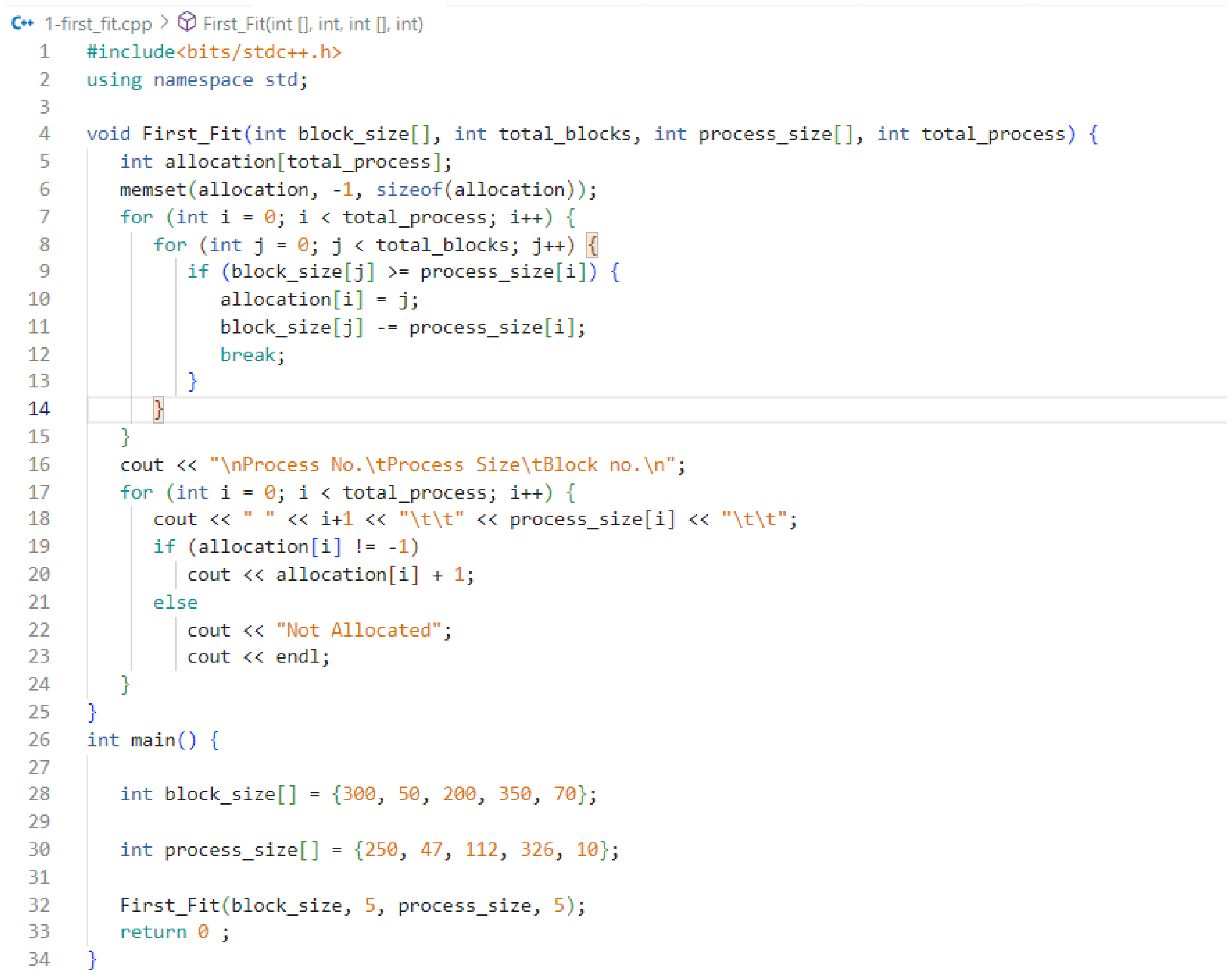
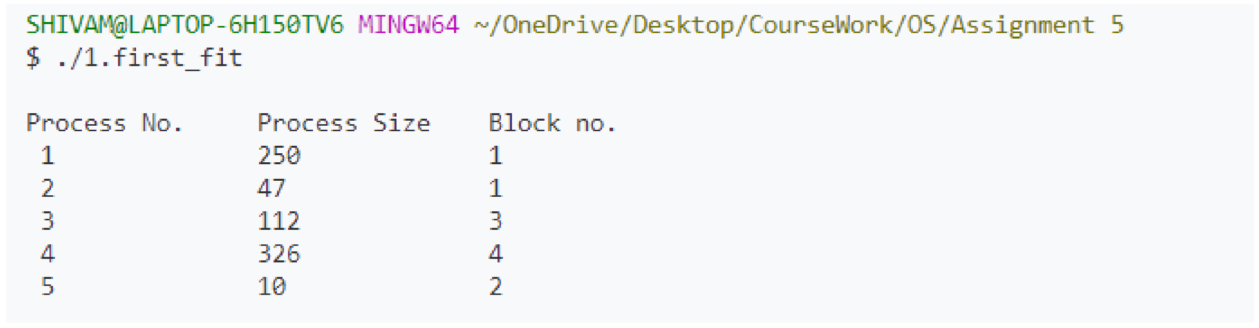
Assignment 5 OS
U20CS135 SHIVAM MISHRA

1.first-fit

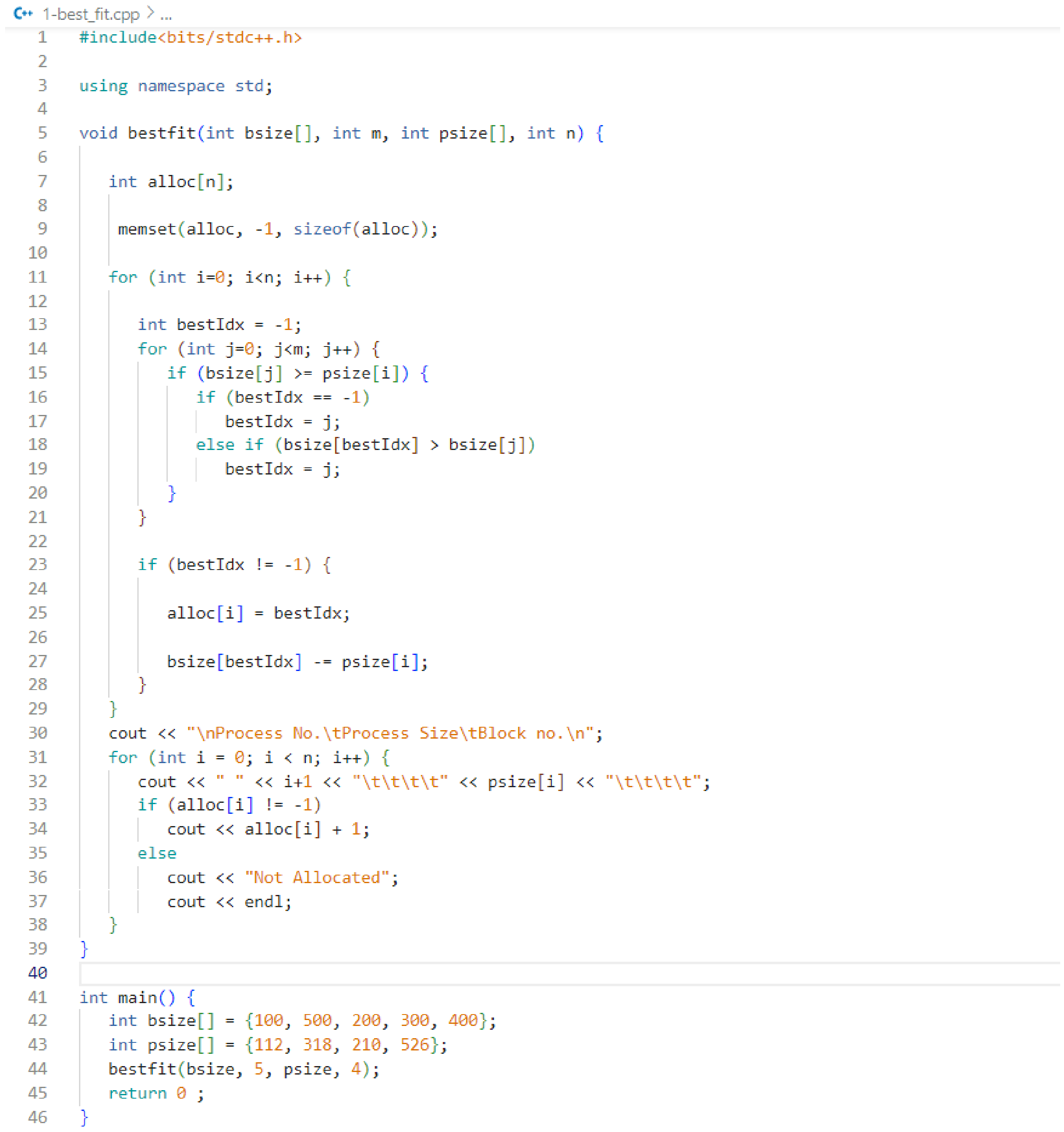


```cpp
#include<bits/stdc++.h>
using namespace std;

void First_Fit(int block_size[], int total_blocks, int process_size[], int total_process) {
   int allocation[total_process];
   memset(allocation, -1, sizeof(allocation));
   for (int i = 0; i < total_process; i++) {
      for (int j = 0; j < total_blocks; j++) {
         if (block_size[j] >= process_size[i]) {
            allocation[i] = j;
            block_size[j] -= process_size[i];
            break;
         }
      }
   }
   cout << "\nProcess No.\tProcess Size\tBlock no.\n";
   for (int i = 0; i < total_process; i++) {
      cout << " " << i+1 << "\t\t" << process_size[i] << "\t\t";
      if (allocation[i] != -1)
         cout << allocation[i] + 1;
      else
         cout << "Not Allocated";
         cout << endl;
   }
}
int main() {

   int block_size[] = {300, 50, 200, 350, 70};

   int process_size[] = {250, 47, 112, 326, 10};

   First_Fit(block_size, 5, process_size, 5);
   return 0 ;
}
```



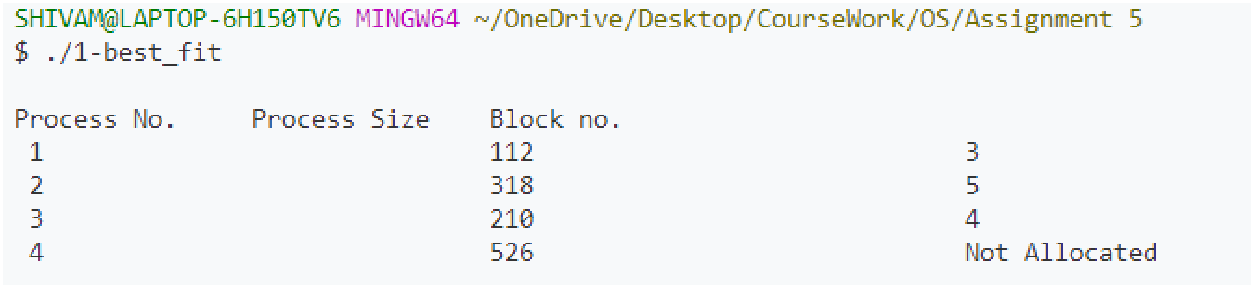
```
SHIVAM@LAPTOP-6H150TV6 MINGW64 ~/OneDrive/Desktop/CourseWork/OS/Assignment 5
$ ./1.first_fit

Process No.     Process Size    Block no.
 1              250             1
 2              47              1
 3              112             3
 4              326             4
 5              10              2
```

1-best-fit

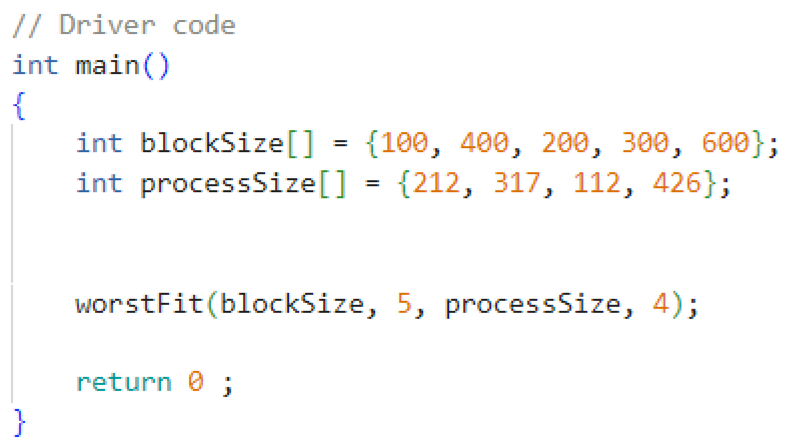


1-best_fit.cpp > ...

```cpp
#include<bits/stdc++.h>

using namespace std;

void bestfit(int bsize[], int m, int psize[], int n) {

    int alloc[n];

     memset(alloc, -1, sizeof(alloc));

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

        int bestIdx = -1;
        for (int j=0; j<m; j++) {
            if (bsize[j] >= psize[i]) {
                if (bestIdx == -1)
                    bestIdx = j;
                else if (bsize[bestIdx] > bsize[j])
                    bestIdx = j;
            }
        }

        if (bestIdx != -1) {

            alloc[i] = bestIdx;

            bsize[bestIdx] -= psize[i];
        }
    }
    cout << "\nProcess No.\tProcess Size\tBlock no.\n";
    for (int i = 0; i < n; i++) {
        cout << " " << i+1 << "\t\t\t\t" << psize[i] << "\t\t\t\t";
        if (alloc[i] != -1)
            cout << alloc[i] + 1;
        else
            cout << "Not Allocated";
            cout << endl;
    }
}

int main() {
    int bsize[] = {100, 500, 200, 300, 400};
    int psize[] = {112, 318, 210, 526};
    bestfit(bsize, 5, psize, 4);
    return 0 ;
}
```


```
SHIVAM@LAPTOP-6H150TV6 MINGW64 ~/OneDrive/Desktop/CourseWork/OS/Assignment 5
$ ./1-best_fit

Process No.     Process Size    Block no.
 1                              112                             3
 2                              318                             5
 3                              210                             4
 4                              526                             Not Allocated
```

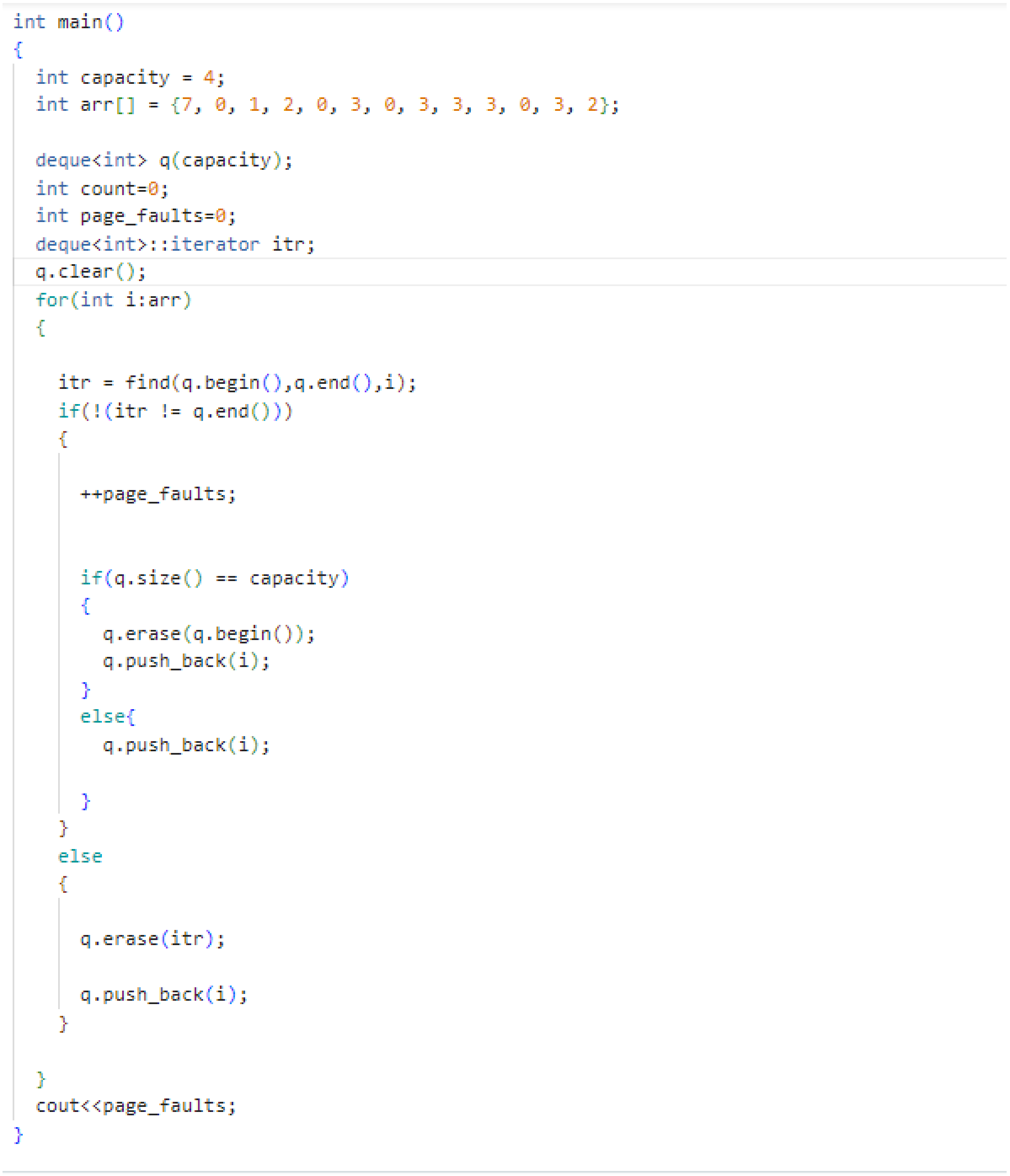
1-worst-fit


```
// Driver code
int main()
{
    int blockSize[] = {100, 400, 200, 300, 600};
    int processSize[] = {212, 317, 112, 426};


    worstFit(blockSize, 5, processSize, 4);

    return 0 ;
}
```

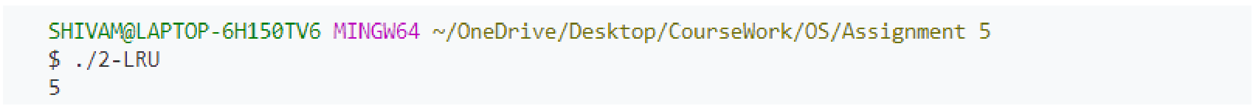
```cpp
void worstFit(int blockSize[], int m, int processSize[],
                                                 int n)
{


    int allocation[n];


    memset(allocation, -1, sizeof(allocation));


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

        int wstIdx = -1;
        for (int j=0; j<m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (wstIdx == -1)
                    wstIdx = j;
                else if (blockSize[wstIdx] < blockSize[j])
                    wstIdx = j;
            }
        }


        if (wstIdx != -1)
        {

            allocation[i] = wstIdx;


            blockSize[wstIdx] -= processSize[i];
        }
    }

     cout << "\nProcess No.\tProcess Size\tBlock no.\n";
    for (int i = 0; i < n; i++)
    {
        cout << "   " << i+1 << " " << processSize[i] << " ";
        if (allocation[i] != -1)
            cout << allocation[i] + 1;
        else
            cout << "Not Allocated";
        cout << endl;
    }
}
```

```
$ ./1-worst_fit

Process No.     Process Size    Block no.
   1 212 5
   2 317 2
   3 112 5
   4 426 Not Allocated
```

2-LRU

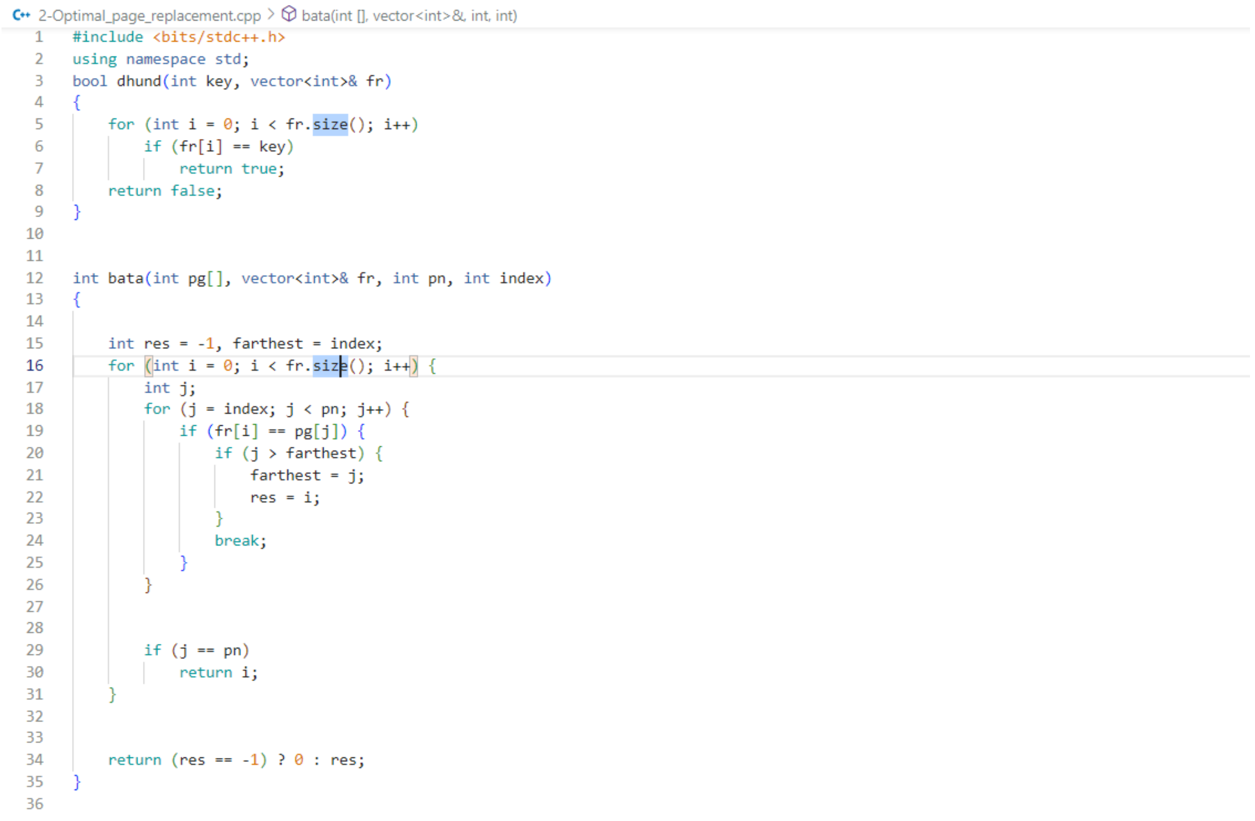


```
int main()
{
  int capacity = 4;
  int arr[] = {7, 0, 1, 2, 0, 3, 0, 3, 3, 3, 0, 3, 2};

  deque<int> q(capacity);
  int count=0;
  int page_faults=0;
  deque<int>::iterator itr;
  q.clear();
  for(int i:arr)
  {

    itr = find(q.begin(),q.end(),i);
    if(!(itr != q.end()))
    {

      ++page_faults;


      if(q.size() == capacity)
      {
        q.erase(q.begin());
        q.push_back(i);
      }
      else{
        q.push_back(i);

      }
    }
    else
    {

      q.erase(itr);

      q.push_back(i);
    }

  }
  cout<<page_faults;
}
```

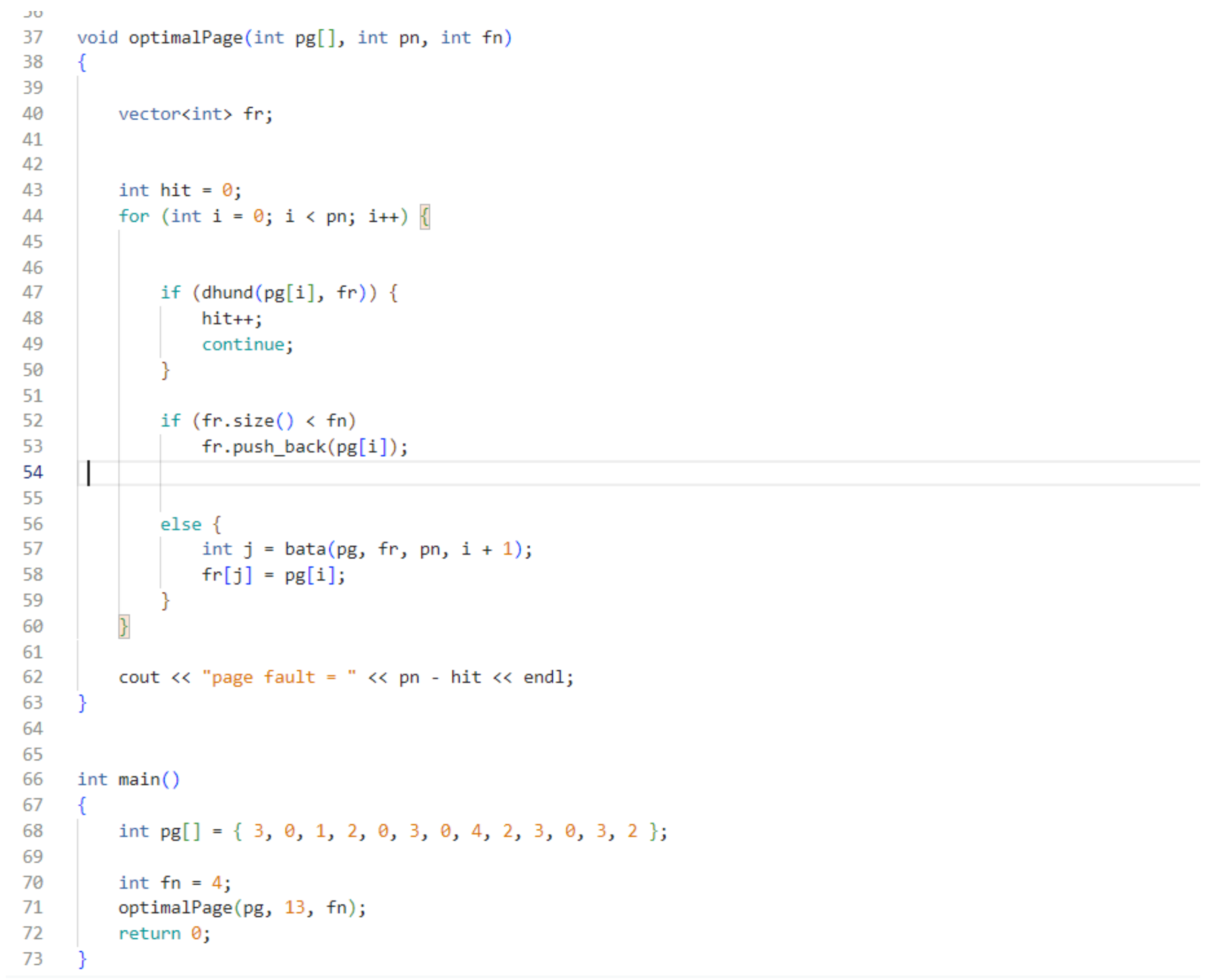


```
SHIVAM@LAPTOP-6H150TV6 MINGW64 ~/OneDrive/Desktop/CourseWork/OS/Assignment 5
$ ./2-LRU
5
```

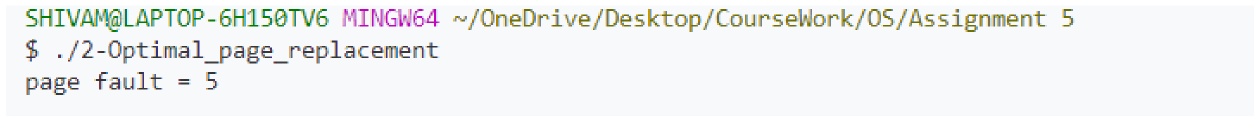
2-optimal page


2-Optimal_page_replacement.cpp > bata(int [], vector<int>&, int, int)

```cpp
#include <bits/stdc++.h>
using namespace std;
bool dhund(int key, vector<int>& fr)
{
    for (int i = 0; i < fr.size(); i++)
        if (fr[i] == key)
            return true;
    return false;
}


int bata(int pg[], vector<int>& fr, int pn, int index)
{

    int res = -1, farthest = index;
    for (int i = 0; i < fr.size(); i++) {
        int j;
        for (j = index; j < pn; j++) {
            if (fr[i] == pg[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }


        if (j == pn)
            return i;
    }


    return (res == -1) ? 0 : res;
}

```



```cpp
void optimalPage(int pg[], int pn, int fn)
{

    vector<int> fr;


    int hit = 0;
    for (int i = 0; i < pn; i++) {


        if (dhund(pg[i], fr)) {
            hit++;
            continue;
        }

        if (fr.size() < fn)
            fr.push_back(pg[i]);


        else {
            int j = bata(pg, fr, pn, i + 1);
            fr[j] = pg[i];
        }
    }

    cout << "page fault = " << pn - hit << endl;
}


int main()
{
    int pg[] = { 3, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 };

    int fn = 4;
    optimalPage(pg, 13, fn);
    return 0;
}
```



```
SHIVAM@LAPTOP-6H150TV6 MINGW64 ~/OneDrive/Desktop/CourseWork/OS/Assignment 5
$ ./2-Optimal_page_replacement
page fault = 5
```