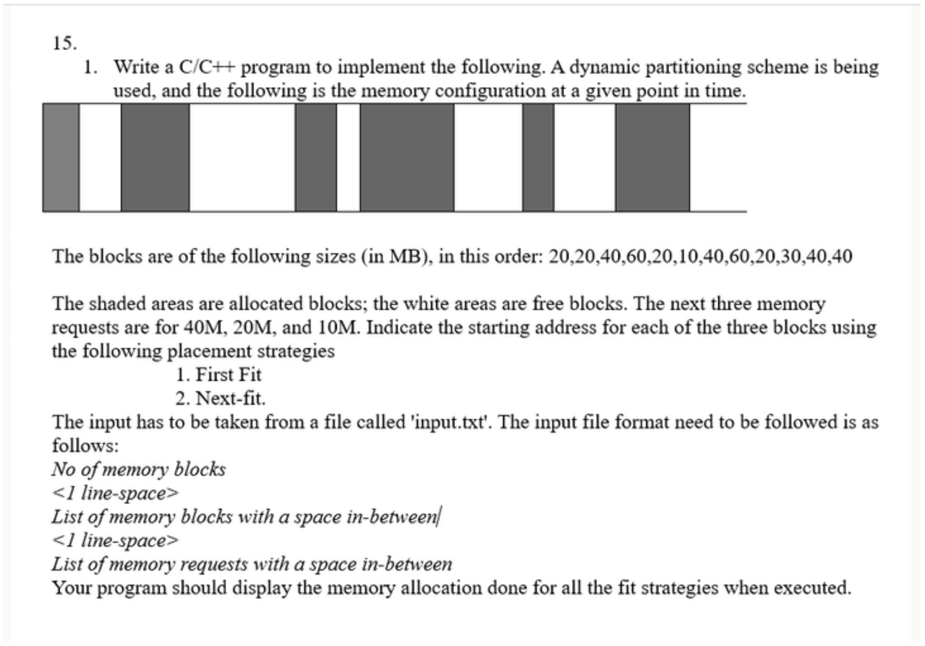
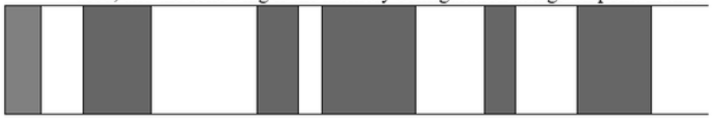
106119029 , OS Lab 10

Dipesh Kafle



15.

1. Write a C/C++ program to implement the following. A dynamic partitioning scheme is being used, and the following is the memory configuration at a given point in time.



The blocks are of the following sizes (in MB), in this order: 20,20,40,60,20,10,40,60,20,30,40,40

The shaded areas are allocated blocks; the white areas are free blocks. The next three memory requests are for 40M, 20M, and 10M. Indicate the starting address for each of the three blocks using the following placement strategies

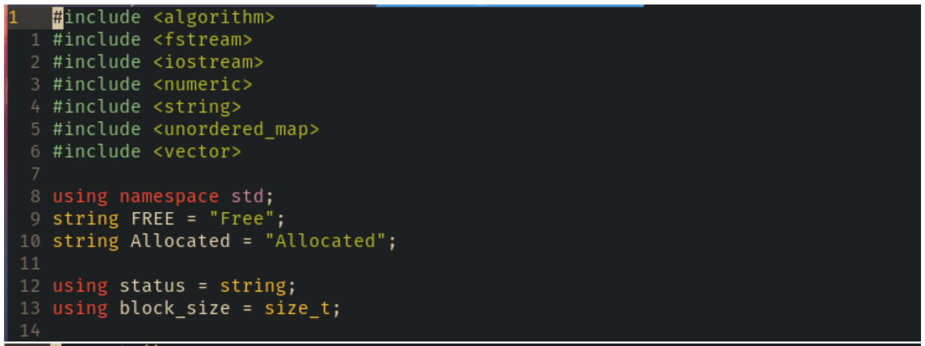
1. First Fit
2. Next-fit.

The input has to be taken from a file called 'input.txt'. The input file format need to be followed is as follows:

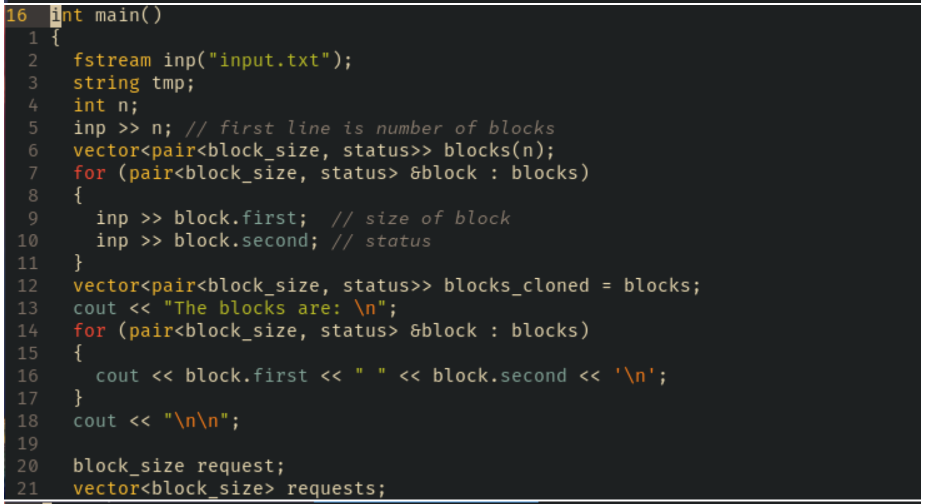
No of memory blocks
<1 line-space>
List of memory blocks with a space in-between
<1 line-space>
List of memory requests with a space in-between

Your program should display the memory allocation done for all the fit strategies when executed.

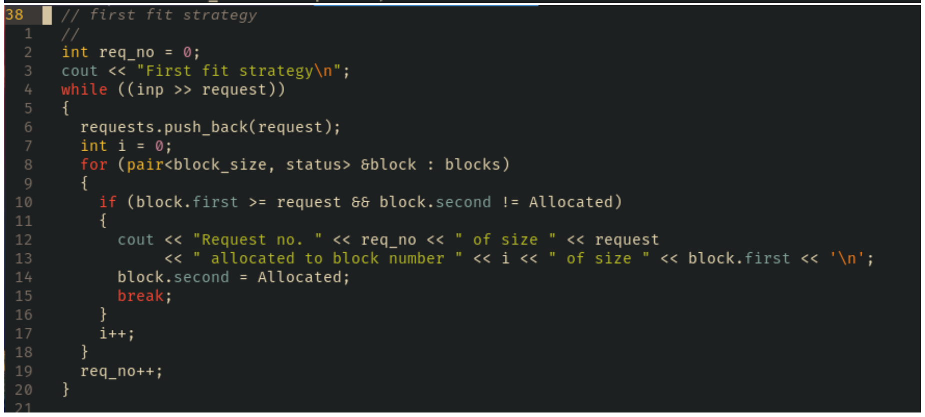
Code



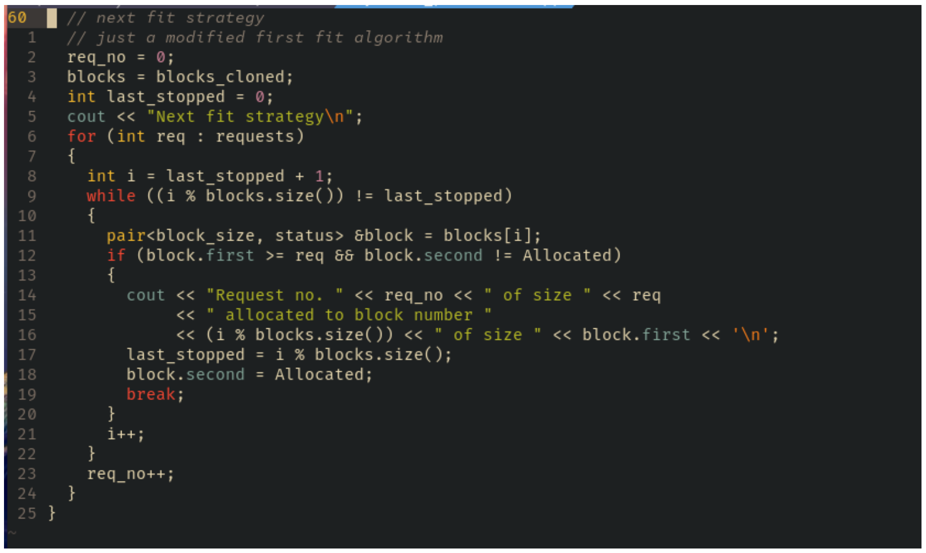
```
#include <algorithm>
#include <fstream>
#include <iostream>
#include <numeric>
#include <string>
#include <unordered_map>
#include <vector>

using namespace std;
string FREE = "Free";
string Allocated = "Allocated";

using status = string;
using block_size = size_t;

```



```
int main()
{
  fstream inp("input.txt");
  string tmp;
  int n;
  inp >> n; // first line is number of blocks
  vector<pair<block_size, status>> blocks(n);
  for (pair<block_size, status> &block : blocks)
  {
    inp >> block.first;  // size of block
    inp >> block.second; // status
  }
  vector<pair<block_size, status>> blocks_cloned = blocks;
  cout << "The blocks are: \n";
  for (pair<block_size, status> &block : blocks)
  {
    cout << block.first << " " << block.second << '\n';
  }
  cout << "\n\n";

  block_size request;
  vector<block_size> requests;
```

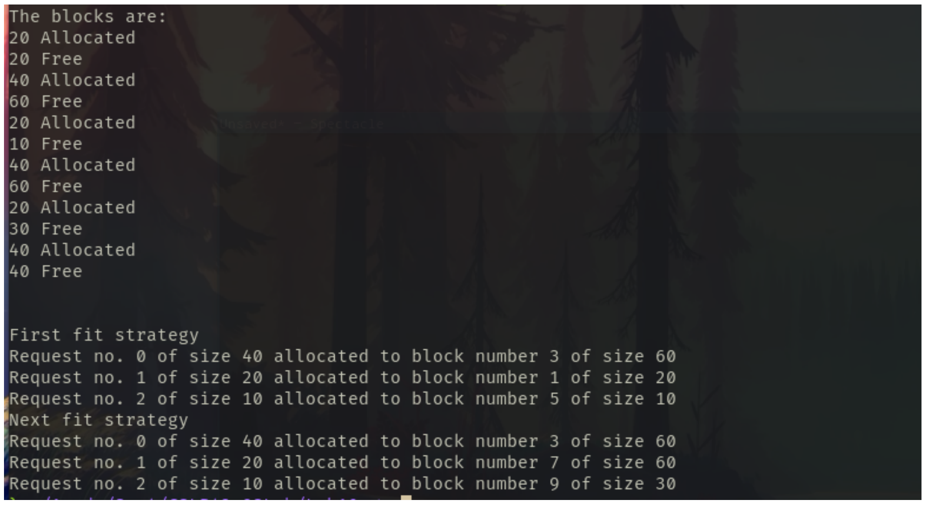


```
// first fit strategy
//
int req_no = 0;
cout << "First fit strategy\n";
while ((inp >> request))
{
  requests.push_back(request);
  int i = 0;
  for (pair<block_size, status> &block : blocks)
  {
    if (block.first >= request && block.second != Allocated)
    {
      cout << "Request no. " << req_no << " of size " << request
           << " allocated to block number " << i << " of size " << block.first << '\n';
      block.second = Allocated;
      break;
    }
    i++;
  }
  req_no++;
}

```



```
// next fit strategy
// just a modified first fit algorithm
req_no = 0;
blocks = blocks_cloned;
int last_stopped = 0;
cout << "Next fit strategy\n";
for (int req : requests)
{
  int i = last_stopped + 1;
  while ((i % blocks.size()) != last_stopped)
  {
    pair<block_size, status> &block = blocks[i];
    if (block.first >= req && block.second != Allocated)
    {
      cout << "Request no. " << req_no << " of size " << req
           << " allocated to block number "
           << (i % blocks.size()) << " of size " << block.first << '\n';
      last_stopped = i % blocks.size();
      block.second = Allocated;
      break;
    }
    i++;
  }
  req_no++;
}
}
```

Output



```
The blocks are:
20 Allocated
20 Free
40 Allocated
60 Free
20 Allocated
10 Free
40 Allocated
60 Free
20 Allocated
30 Free
40 Allocated
40 Free


First fit strategy
Request no. 0 of size 40 allocated to block number 3 of size 60
Request no. 1 of size 20 allocated to block number 1 of size 20
Request no. 2 of size 10 allocated to block number 5 of size 10
Next fit strategy
Request no. 0 of size 40 allocated to block number 3 of size 60
Request no. 1 of size 20 allocated to block number 7 of size 60
Request no. 2 of size 10 allocated to block number 9 of size 30
```