**Program 4**

1. **Problem Statement**

The goal of the project was to analyze four memory partitioning algorithms by running 1000 experiments on each algorithm and print the final average result of how long each algorithm takes to complete. The memory holds 56 units of free memory slots, 1000 tasks of varying sizes and process times are passed into the memory and are processed. The time taken to add tasks to memory and continue looping tasks into memory as tasks are being completed are to be calculated and presented as the output of the program.

1. **Approach to Solution**

A structure is populated with random sizes and process times, and additional elements that will keep track of what tasks are queued, processed, etc. An empty dynamic memory array with 56 blocks is initialized in each algorithm function and the task info stream is passed into each function. The algorithms will use various additional functions to determine memory blocks that will be able to process each task. The best-fit algorithm will use the smallest() function to determine which memory block is small enough to hold a task of its respective size. The worst-fit algorithm will use the biggest() function to determine which memory block is biggest to fit a task of its respective size. A search array function is used to find a block of memory that can hold the first task in the info stream, this is used for both first-fit and next-fit. Additional functions are used to determine time taken for each algorithm and reset info streams to be reanalyzed. I used cLion and developed my program in c++.

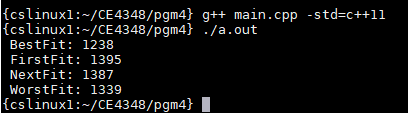
1. **Solution Description**

To run the code in the cs linux terminal use command:

$ g++ main.cpp -std=c++11

$g++ ./a.out

(needed to use -std=c++11 because I used nullptr in my code.)



The program will take a few seconds to display results. The image shows the results I got after running my code and as per the professor’s estimate, the code seems relatively accurate. The best algorithm to use from this experimentation seems to be best-fit since it has the quickest time and has the most optimized approach to placing tasks in memory.