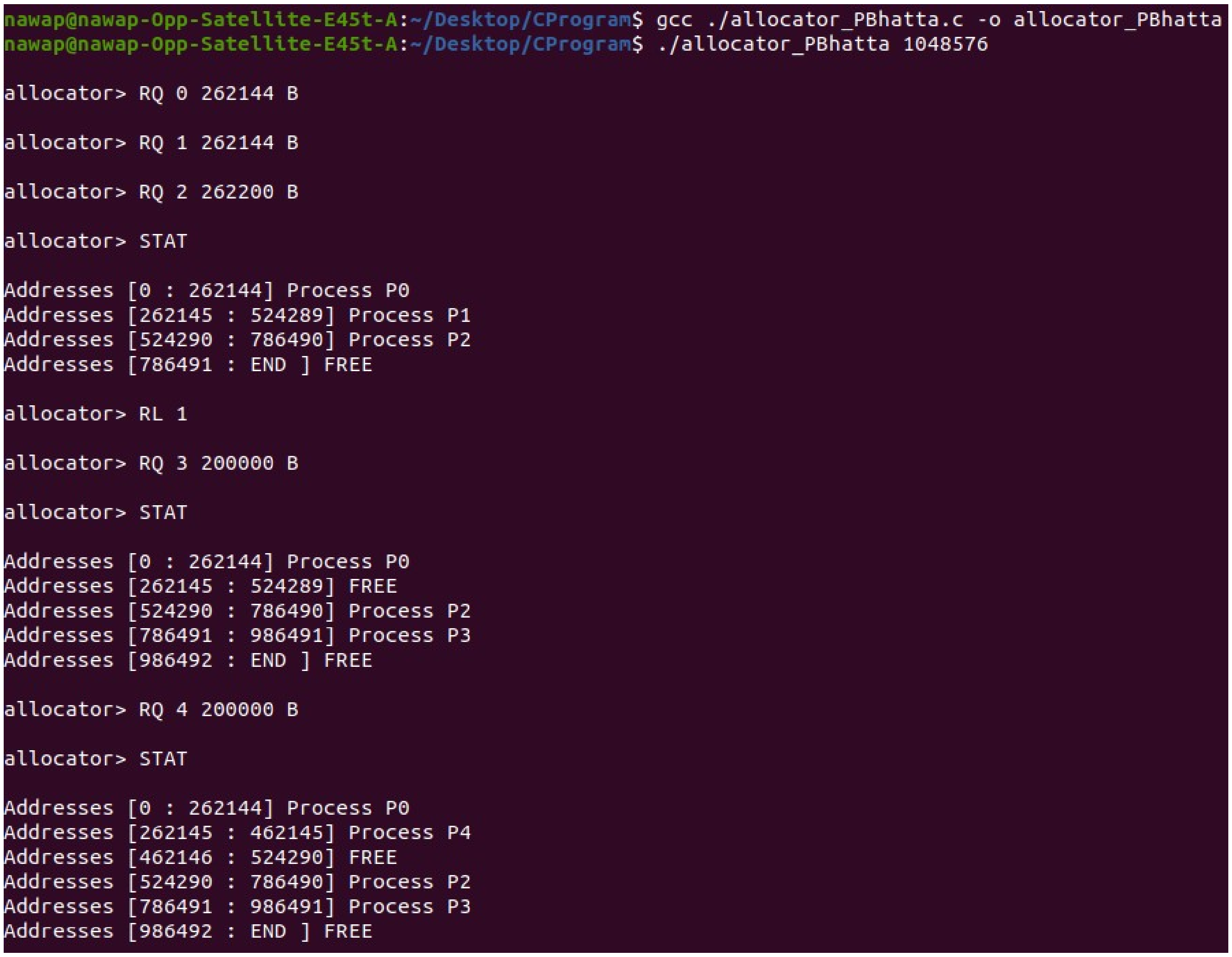
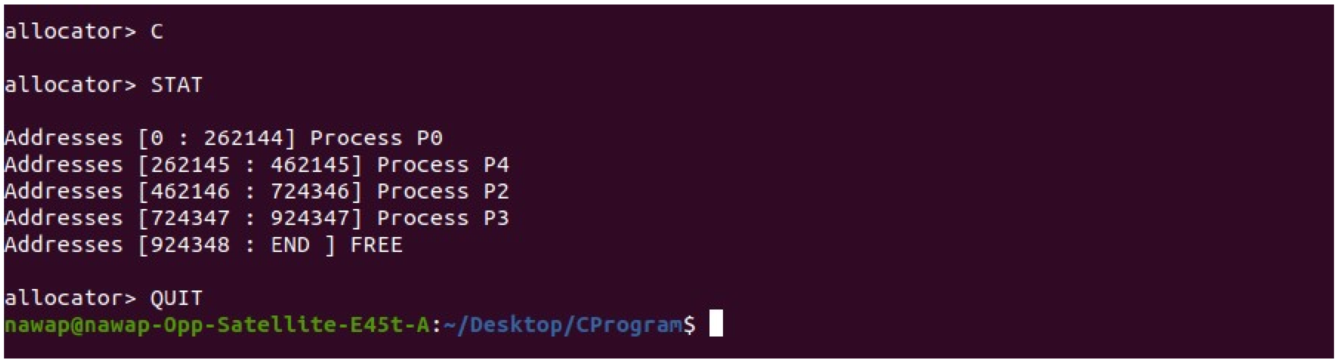
Instruction to compile and generate executable file:
----

1. Download allocator_PBhatta.c.
2. In terminal, type in 'gcc ./allocator_PBhatta.c -o allocator_PBhatta' which should compile the code and generate executable file named 'allocator_PBhatta'.
3. Now run that file by giving memory size in bytes as argument like './allocator_PBhatta 1048576' .
4. 'allocator>' should appear in the console. Now choose from RQ, RL, C, STAT and QUIT commands. RQ will allow making request of certain memory size for specified process_id. RL will release process by process_id. C will do compaction. STAT will show memory space occupied by each processes and also mention which part of memory is empty. QUIT will terminate the program. Please note that process_id is made integer value. Therefore, RQ command will look like: 'RQ 123456 200000 B' where 123456 is process_id.

Output:



```
nawap@nawap-Opp-Satellite-E45t-A:~/Desktop/CProgram$ gcc ./allocator_PBhatta.c -o allocator_PBhatta
nawap@nawap-Opp-Satellite-E45t-A:~/Desktop/CProgram$ ./allocator_PBhatta 1048576

allocator> RQ 0 262144 B

allocator> RQ 1 262144 B

allocator> RQ 2 262200 B

allocator> STAT

Addresses [0 : 262144] Process P0
Addresses [262145 : 524289] Process P1
Addresses [524290 : 786490] Process P2
Addresses [786491 : END ] FREE

allocator> RL 1

allocator> RQ 3 200000 B

allocator> STAT

Addresses [0 : 262144] Process P0
Addresses [262145 : 524289] FREE
Addresses [524290 : 786490] Process P2
Addresses [786491 : 986491] Process P3
Addresses [986492 : END ] FREE

allocator> RQ 4 200000 B

allocator> STAT

Addresses [0 : 262144] Process P0
Addresses [262145 : 462145] Process P4
Addresses [462146 : 524290] FREE
Addresses [524290 : 786490] Process P2
Addresses [786491 : 986491] Process P3
Addresses [986492 : END ] FREE
```



```
allocator> C

allocator> STAT

Addresses [0 : 262144] Process P0
Addresses [262145 : 462145] Process P4
Addresses [462146 : 724346] Process P2
Addresses [724347 : 924347] Process P3
Addresses [924348 : END ] FREE

allocator> QUIT
nawap@nawap-Opp-Satellite-E45t-A:~/Desktop/CProgram$ 
```

General Discussion On The Code
----
1. **Background:** This project took me about couple days to complete. During that time, I learned a lot about memory and memory management. I have also learned to appreciate the working mechanism of memory management although it looks simple from the surface view.

2. **Objective and Design of the Project:** The objective of the program is to implement best fit strategy in memory management. It also does compaction in order to remove smaller chunks of empty spaces created after deletion of processes. Since at worse case, there could be fragment of empty space after every process block, compaction becomes necessary element in memory management to free memory space to accommodate more incoming processes.

3. General Algorithm on how the code was formulated:
I) First user input is taken to instantiate total memory space available for the program.
II) Then user input is taken in order to allocate memory space for certain process. That process can be deleted using RL command.
III) When process comes in, it does not enter straight into first available block of memory. Instead, computation is done in such a way that process will enter the best block with minimum possible space. We compare all the fitting blocks and choose the smallest one among them. This is also called Best Fit.
IV) Release or Deletion is formulated by simply putting a flag of empty in the block that is to be released.
V) Compaction is done by creating entirely new block of memory where processes are brought in. We ignore empty blocks in the middle. But we sum the total spaces of such blocks.
VI) At the end, the empty single block of size that equals sum of empty fragments is added.

4. **Some Challenges:**
The first challenge that I faced was about creating a program that takes in commands one after other without ending the process. I put the program in while(true) loop and allowed program to terminate only when QUIT is entered. I also designed the code in such a way that it handles most of the errors in input. That was also a challenge.
The second challenge I faced was while writing best fit algorithm. I did it in linked list therefore it was kind of tedious process. I had to traverse the list several times just to accomplish a single objective. But ultimately, I got it working.
The third and the ultimate challenge for me would be compaction algorithm. Though it looks simple from the surface, it involves lots of things to consider. I did it in linked list. So I had to traverse the list so many times and take care of so many things all at once. I first tried doing it by deleting the empty fragments in the middle and updating the next block’s base address. That didn’t quite work out as I thought it would. Therefore, I had to come up with entirely fresh plan. I created a brand new list where I would take only blocks that are filled with processses one after another. Then at the end of the new list, I added a block marked as empty that has total sum of fragmented spaces.