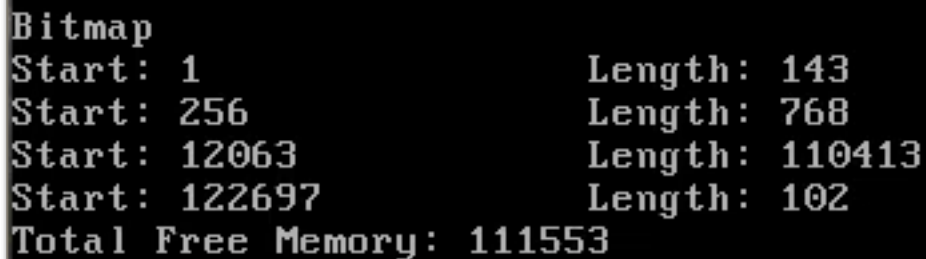


In order to prove that we implemented first fit properly, we added printout statements to execute after the bitmap was changed. The printout statements illustrated where the holes in memory started, and how long they were. Holes that had a length of less than 100 were ignored.

An example of this output is below.



```
Bitmap
Start: 1           Length: 143
Start: 256         Length: 768
Start: 12063        Length: 110413
Start: 122697       Length: 102
Total Free Memory: 111553
```

We then used the provided forkmem function and associated memory-usage functions to view the memory usage as forkmem executed.

The forkmem command used was **`.forkmem 12+0+4 40+2+50 3+50+3`**

For this output, we expected a memory usage similar to:

```
  | 12 |
| 40 | 12 |
  | 40 |
| 3 | 40 |
```

for the next-fit implementation (reverse order because the findbits() function starts scanning at the largest possible memory address), and:

```
  | 12 |
| 40 | 12 |
  | 40 |
| 40 | | 3 |
```

for the first-fit implementation. (Consecutive lines represent processes as they are started and terminated).

When we recorded the printout statements with a screen capture, we saw this functionality for our algorithm, so we concluded that it was correct.