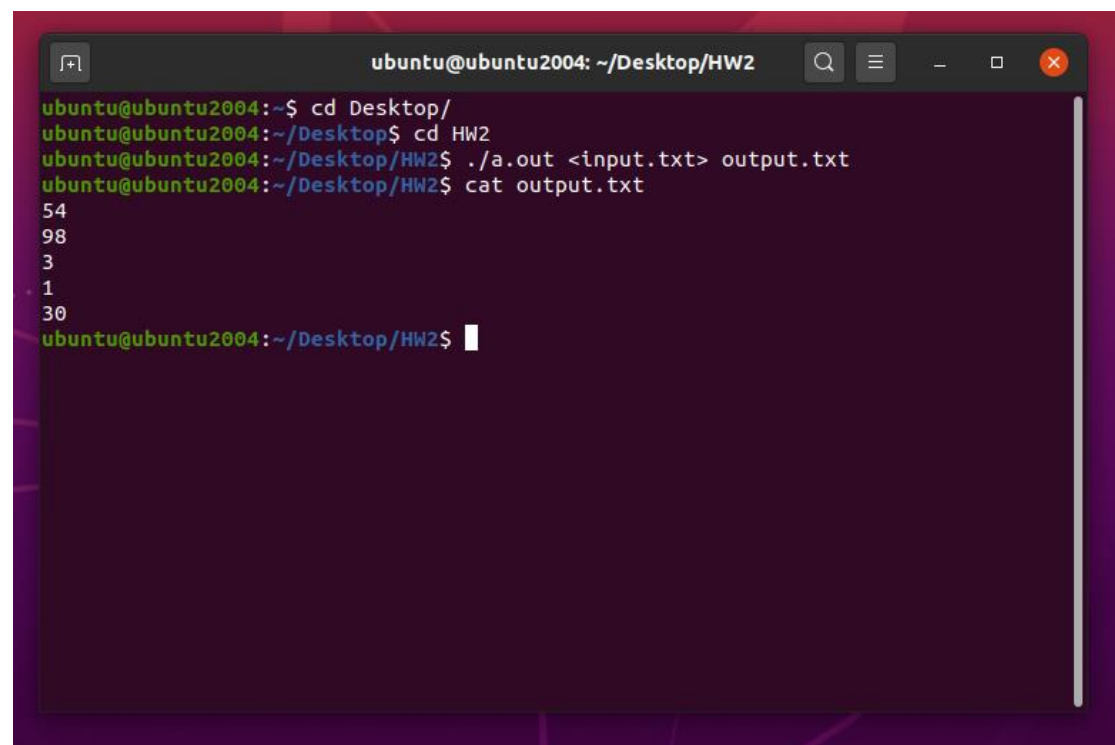


Result screenshot:

A terminal window titled 'ubuntu@ubuntu2004: ~/Desktop/HW2' with search, menu, and window control icons. The terminal shows the following commands and output:

```
ubuntu@ubuntu2004:~$ cd Desktop/  
ubuntu@ubuntu2004:~/Desktop$ cd HW2  
ubuntu@ubuntu2004:~/Desktop/HW2$ ./a.out <input.txt> output.txt  
ubuntu@ubuntu2004:~/Desktop/HW2$ cat output.txt  
54  
98  
3  
1  
30  
ubuntu@ubuntu2004:~/Desktop/HW2$
```

Program architecture:

先將每行指令逐一讀入，再根據不同的指令，使用 if else 判斷去個別處理，最終將答案寫入 output.txt 內，就完成了(詳細說明在 how you design your program 與 functions)

我是用 switch 與 if else 搭配使用來做事的。這次的開檔不同於以往，這次是用 bash redirection 的方式做的。

Program functions:

1. 自定義函式:int power(int a,int b)

這個自定義函式用來處理 PUSH 指令後面的 random index 的，將字元轉換成 int 型態去儲存的時候，由於不同位數，相當於該位數的數字乘上 10 的不同次方，為了縮短程式碼並簡化，使其能寫在同一個 for 回圈內就解決，所以自定義了這個 function。a 代表底數，b 代表指數，以下是這個副程式的內容:

```
int power(int a,int b)  
{  
    int ans=1;  
    if(b<1)  
    {  
        return ans;  
    }else
```

```

    {
        for(int i=1;i<=b;i++)
        {
            ans*=a;
        }
        return ans;
    }
}

```

```
2. int sprintf(char *str, const char *format, ...)
```

這個函式是我在將 `int` 型態的數字轉為 `string` 的時候用的，因為必須先將數字轉為 `char` 或是 `char array`，才能用 `fputs` 或是 `fputc` 寫入 `output.txt` 裡面，我原本打算自己寫一個副程式來處理 `int` 轉 `string`，可是上網查詢後發現有這個好用的函式可以直接幫我處理。我可以先設一個 `char` 型態的陣列 `c`，然後這樣做：
`sprintf(c,"%d",需要被我轉換成字串的某個 int 型態的整數)`，就可以直接將該整數直接轉成字串型態並存入字元陣列 `c` 裡面。

```
3. char *strchr(const char *str, int c)
```

這個函式我為了處理 `Unix` 與 `windows` 與 `MacOs` 三種不同的 `OS` 底下，換行符號的不同而使用的函式，先傳入某個字元陣列的開頭位址，然後再寫入我需要搜尋的目標字元，這個函式就可以幫我找到字串中的目標字元的位址，並回傳那個位址給我，我就可以對目標字元進行修改或處理。

4.how you design your program

這次的作業是要我們模擬 `stack` 與 `queue`，觀察作業說明可以發現，那疊盤子填充的順序與客人拿取的順序是相反的，比如盤子填充的順序是 `30 98 54`，但客人拿取盤子的順序是 `54 98 30`，這就是堆疊的概念，先進後出，後進先出，所以我在這個部分，我使用一個 `plate` 陣列去存放盤子的編號，但是我採用的方式不是按照盤子進來的先後順序，從陣列第 `0` 格、第一格這樣排，我是將每次 `PUSH` 的盤子編號都往陣列的最後方丟，然後每次 `POP` 再從陣列最前方開始取，這樣就可以模擬 `stack` 的感覺。

我一開始初始化的時候就將每格 `plate` 陣列的值設為 `0`，有新的盤子進入該格，再更改該格的值，這邊有一個點就是，若盤子被 `POP` 走了，必須將該格重新設為 `0`。我還有設一個 `plate1` 變數去存目前 `plate` 陣列中盤子堆疊的高度，原始值為 `10000`，每 `PUSH` 一個盤子就-1，比如目前 `PUSH` 了三個盤子進來，則 `plate1` 值為 `9997`，代表目前自陣列第 `9997` 格開始有盤子，若盤子被 `POP` 走了，則該格歸 `0` 並將 `plate1` 值+1，比如剛剛的情況，若 `POP` 一個盤子，則

plate1 值為 9998，代表陣列自第 9998 格開始才有盤子。

接下來我發現一件事，每次是 A 還是 B 去 POP，是由下一行，哪條線 ENQUEUE 去決定的，所以我在 while 裡面，每次讀到那行，只要是 POP，就 continue，直到遇到 ENQUEUE 再去處理實際上是 POP 的動作。