game.c:

1. 向系統 catch　SIGUSR1 信號和 SIGINT 信號，並分別指向 Check_handler 與 sigint_handler.
2. 利用 shmget() 創建或取得一個大小為 struct data 共享記憶體區段，並取得其 ID。
3. 根據 ID 利用 shmat()取得共享記憶體區段的指標位址，並進行操作。
4. 當 接收到 SIGUSR1 信號時，呼叫 Check_handler 將 shared memory 中的 guess 變數與被猜的數字做比較，並將結果(smaller、bigger、bingo)寫回 result 變數當中。
5. 程式結束時 (Ctrl + C)，接收到 SIGINT 信號，利用 shmdt()分離共享記憶體，最後利用 shmctl() 刪除(釋放)共享記憶體段。

guess.c:

1. 利用 key 透過 shmget() 取得 ID 與 game.c 一致的共享記憶體區段。
2. 用 shmat()取得共享記憶體區段的指標位址，並進行操作。
3. 向系統 catch　SIGALRM 信號，並指向 guess_handler.
4. 設定計時器的初始值為一秒，計時器的重製間隔也為一秒，並開始計時，每秒發送一次 SIGALRM，並呼叫 guess_handler.
5. 呼叫 guess_handler 後，讀取 result 變數的結果，計算該回合要猜的數值並將其寫入 guess 變數當中，最後利用 kill(pid, SIGUSR1); 發送 SIGUSR1 信號到 process ID 為 pid 的行程。
6. 一開始 result 變數並無任何資訊因此 guess=(upper_bound+lower_bound)/2，接下來若 result 變數為 smaller，則 upper_bound 變為上一輪所猜的數字減一(upper_bound=guess-1),接著計算 guess=(upper_bound+lower_bound)/2，若 result 變數為 bigger，則 lower_bound 變為上一輪所猜的數字加一(lower_bound=guess+1)，若 result 變數為 bingo，則利用 shmdt()分離共享記憶體並結束程式。