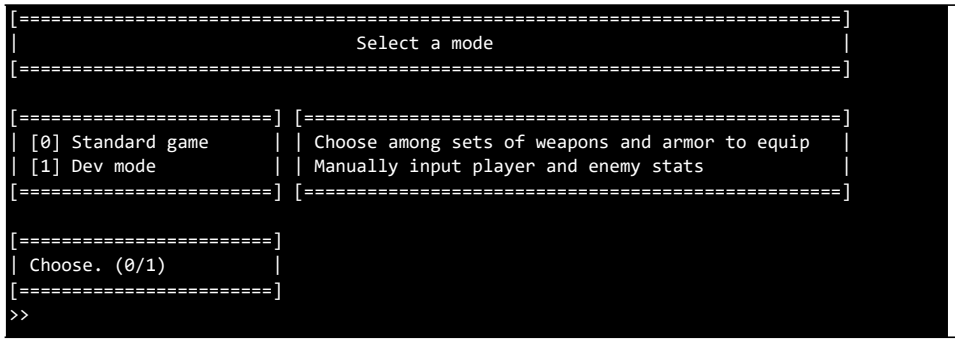
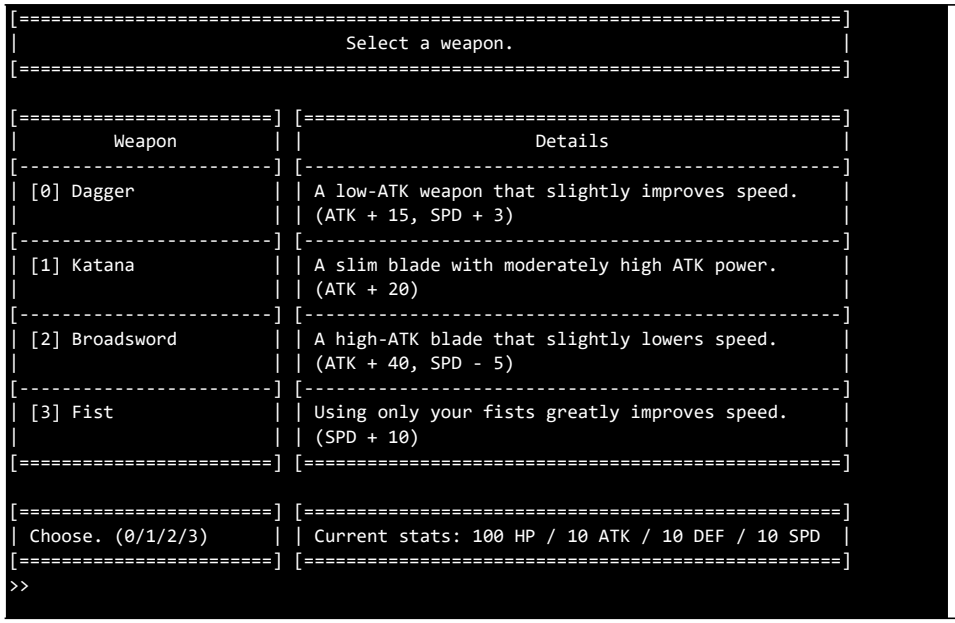
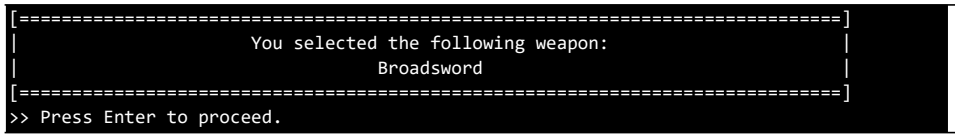
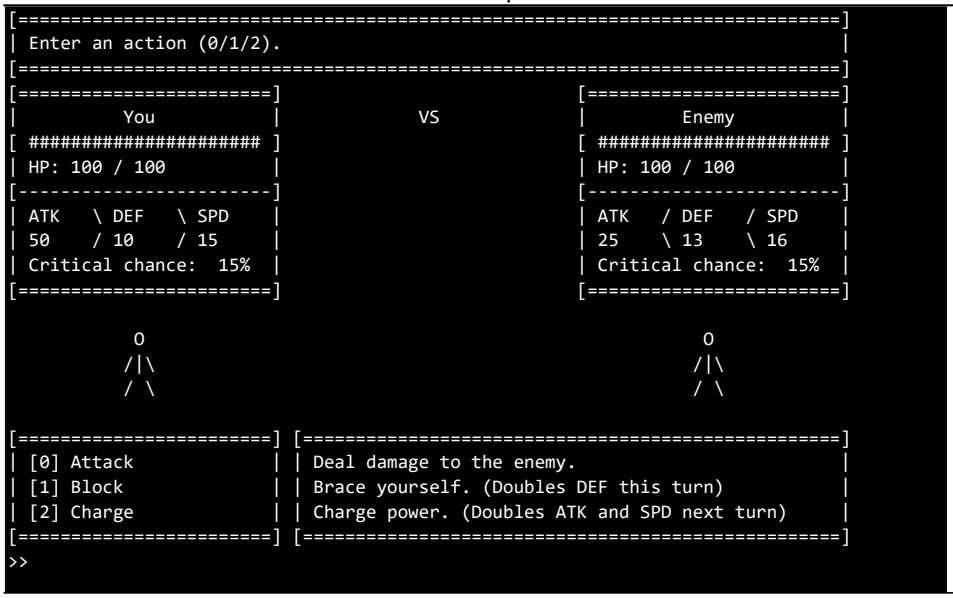
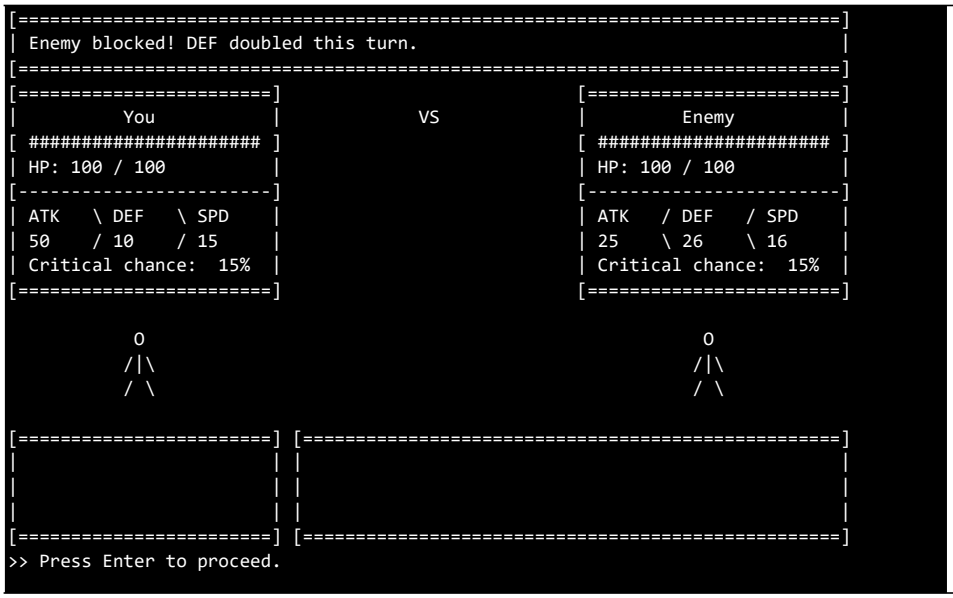
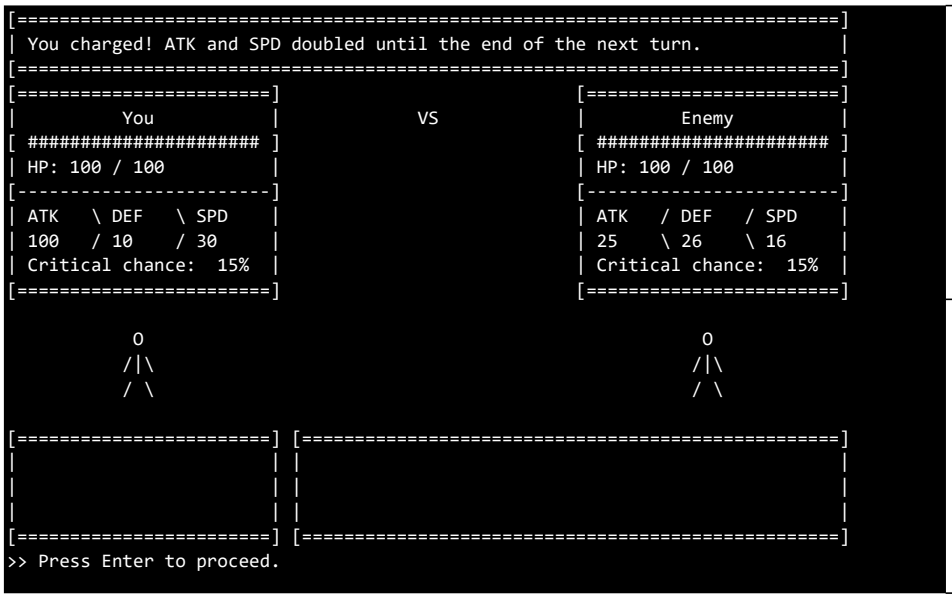
Function name	#	Test description	Sample input			Expected result			Actual result			Pass / fail
			Type	Name	Value / value at address	Type	Name	Value / value at address	Type	Parameter name	Value / value at address	
int promptSelectMode **Input conditions:** nInput is from scanf() nInput is 0 or 1	1	scanf() nInput < 0	**int**	nInput	scanf() => -1	Invalid input for nInput			Invalid input for nInput			Pass
		scanf() nInput = 0	**int**	nInput	scanf() => 0	**int**	***function return***	0	**int**	***function return***	0	Pass
		scanf() nInput = 1	**int**	nInput	scanf() => 1	**int**	***function return***	1	**int**	***function return***	1	Pass
		scanf() nInput > 1	**int**	nInput	scanf() => 2	Invalid input for nInput			Invalid input for nInput			Pass
void promptSetStats **Input conditions:** Inputs are from scanf() ***nTargetHP**, ***nTargetATK**, ***nTargetDEF**, and ***nTargetSPD** are less than or equal to 999 ***nTargetCrit** is less than or equal to 100	2	scanf() 100/10/10/10/15	**int ***	nTargetHP	scanf() => 100	**int ***	nTargetHP	100	**int ***	nTargetHP	100	Pass
			int *	nTargetATK	scanf() => 10	**int ***	nTargetATK	10	**int ***	nTargetATK	10	Pass
			int *	nTargetDEF	scanf() => 10	**int ***	nTargetDEF	10	**int ***	nTargetDEF	10	Pass
			int *	nTargetSPD	scanf() => 10	**int ***	nTargetSPD	10	**int ***	nTargetSPD	10	Pass
			int *	nTargetCrit	scanf() => 15	**int ***	nTargetCrit	15	**int ***	nTargetCrit	15	Pass
		scanf() 1000/20/20/20/100	**int ***	nTargetHP	scanf() => 1000	Invalid input for *nTargetHP			Invalid input for *nTargetHP			Pass
			int *	nTargetATK	scanf() => 20	**int ***	nTargetATK	20	**int ***	nTargetATK	20	Pass
			int *	nTargetDEF	scanf() => 20	**int ***	nTargetDEF	20	**int ***	nTargetDEF	20	Pass
			int *	nTargetSPD	scanf() => 20	**int ***	nTargetSPD	20	**int ***	nTargetSPD	20	Pass
			int *	nTargetCrit	scanf() => 100	**int ***	nTargetCrit	100	**int ***	nTargetCrit	100	Pass
		scanf() 999/999/999/999/999	**int ***	nTargetHP	scanf() => 999	**int ***	nTargetHP	999	**int ***	nTargetHP	999	Pass
			int *	nTargetATK	scanf() => 999	**int ***	nTargetATK	999	**int ***	nTargetATK	999	Pass
			int *	nTargetDEF	scanf() => 999	**int ***	nTargetDEF	999	**int ***	nTargetDEF	999	Pass
			int *	nTargetSPD	scanf() => 999	**int ***	nTargetSPD	999	**int ***	nTargetSPD	999	Pass
			int *	nTargetCrit	scanf() => 999	Invalid input for *nTargetCrit			Invalid input for *nTargetCrit			Pass
char * equipWeapon **Preconditions:** **nInput** is 0, 1, 2, or 3 ***nTargetATK** and ***nTargetSPD** are non-negative and less than or equal to 999	3	nInput = 0	**int ***	nTargetATK	10	**int ***	nTargetATK	25	**int ***	nTargetATK	25	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	13	**int ***	nTargetSPD	13	Pass
			int	nInput	0	**char ***	***function return***	"Dagger"	**char ***	***function return***	"Dagger"	Pass
		nInput = 1	**int ***	nTargetATK	10	**int ***	nTargetATK	30	**int ***	nTargetATK	30	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	10	**int ***	nTargetSPD	10	Pass
			int	nInput	1	**char ***	***function return***	"Katana"	**char ***	***function return***	"Katana"	Pass
		nInput = 2	**int ***	nTargetATK	10	**int ***	nTargetATK	50	**int ***	nTargetATK	50	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	5	**int ***	nTargetSPD	5	Pass
			int	nInput	2	**char ***	***function return***	"Broadsword"	**char ***	***function return***	"Broadsword"	Pass
		nInput = 3	**int ***	nTargetATK	10	**int ***	nTargetATK	10	**int ***	nTargetATK	10	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	20	**int ***	nTargetSPD	20	Pass
			int	nInput	3	**char ***	***function return***	"Fist"	**char ***	***function return***	"Fist"	Pass
char * equipArmor **Preconditions:** **nInput** is 0, 1, 2, or 3 ***nTargetHP**, ***nTargetDEF**, and ***nTargetSPD** are non-negative and less than or equal to 999	4	nInput = 0	**int ***	nTargetHP	100	**int ***	nTargetHP	100	**int ***	nTargetHP	100	Pass
			int *	nTargetDEF	10	**int ***	nTargetDEF	13	**int ***	nTargetDEF	13	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	13	**int ***	nTargetSPD	13	Pass
			int	nInput	0	**char ***	***function return***	"Mythril"	**char ***	***function return***	"Mythril"	Pass
		nInput = 1	**int ***	nTargetHP	100	**int ***	nTargetHP	105	**int ***	nTargetHP	105	Pass
			int *	nTargetDEF	10	**int ***	nTargetDEF	18	**int ***	nTargetDEF	18	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	10	**int ***	nTargetSPD	10	Pass
			int	nInput	1	**char ***	***function return***	"Chainmail"	**char ***	***function return***	"Chainmail"	Pass
		nInput = 2	**int ***	nTargetHP	100	**int ***	nTargetHP	110	**int ***	nTargetHP	110	Pass
			int *	nTargetDEF	10	**int ***	nTargetDEF	28	**int ***	nTargetDEF	28	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	6	**int ***	nTargetSPD	6	Pass
			int	nInput	2	**char ***	***function return***	"Adamantite armor"	**char ***	***function return***	"Adamantite armor"	Pass
		nInput = 3	**int ***	nTargetHP	100	**int ***	nTargetHP	100	**int ***	nTargetHP	100	Pass
			int *	nTargetDEF	10	**int ***	nTargetDEF	2	**int ***	nTargetDEF	2	Pass
			int *	nTargetSPD	10	**int ***	nTargetSPD	20	**int ***	nTargetSPD	20	Pass
			int	nInput	3	**char ***	***function return***	"No armor"	**char ***	***function return***	"No armor"	Pass
int promptWeaponSelect **Input conditions:** nInput is from scanf() nInput is 0, 1, 2, or 3	5	scanf() nInput < 0	**int**	nInput	scanf() => -1	Invalid input for nInput			Invalid input for nInput			Pass
		scanf() nInput = 0	**int**	nInput	scanf() => 0	**int**	***function return***	0	**int**	***function return***	0	Pass
		scanf() nInput = 1	**int**	nInput	scanf() => 1	**int**	***function return***	1	**int**	***function return***	1	Pass
		scanf() nInput = 2	**int**	nInput	scanf() => 2	**int**	***function return***	2	**int**	***function return***	2	Pass
		scanf() nInput = 3	**int**	nInput	scanf() => 3	**int**	***function return***	3	**int**	***function return***	3	Pass
		scanf() nInput > 3	**int**	nInput	scanf() => 4	Invalid input for nInput			Invalid input for nInput			Pass

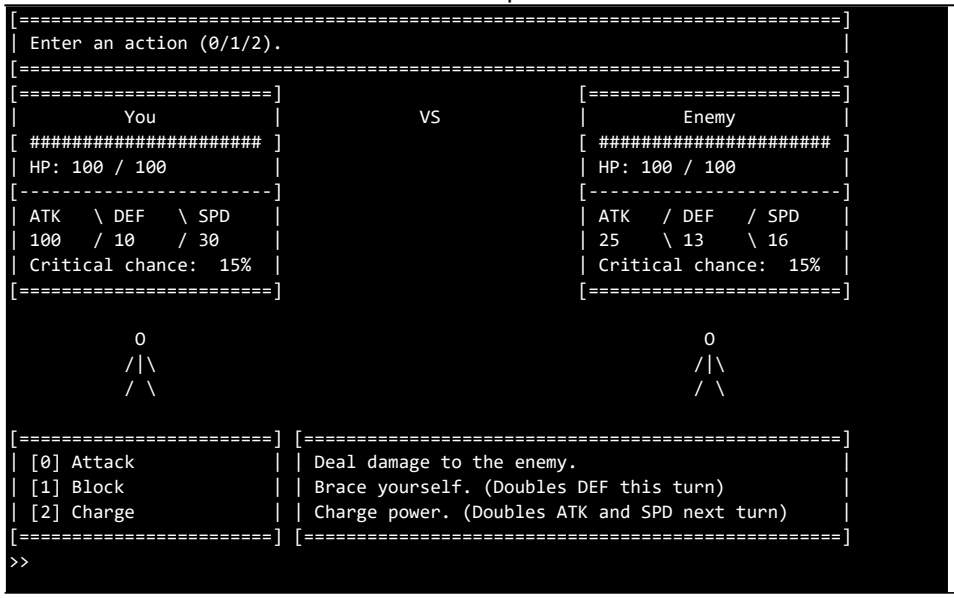
Function name	#	Test description	Sample input			Expected result			Actual result			Pass / fail
			Type	Name	Value / value at address	Type	Name	Value / value at address	Type	Parameter name	Value / value at address	
int promptArmorSelect **Input conditions:** nInput is from scanf() nInput is 0, 1, 2, or 3	6	scanf() nInput < 0	**int**	nInput	scanf() => -1	Invalid input for nInput			Invalid input for nInput			Pass
		scanf() nInput = 0	**int**	nInput	scanf() => 0	**int**	*function return*	0	**int**	*function return*	0	Pass
		scanf() nInput = 1	**int**	nInput	scanf() => 1	**int**	*function return*	1	**int**	*function return*	1	Pass
		scanf() nInput = 2	**int**	nInput	scanf() => 2	**int**	*function return*	2	**int**	*function return*	2	Pass
		scanf() nInput = 3	**int**	nInput	scanf() => 3	**int**	*function return*	3	**int**	*function return*	3	Pass
		scanf() nInput > 3	**int**	nInput	scanf() => 4	Invalid input for nInput			Invalid input for nInput			Pass
int promptPlayerEquipConfirm **Input conditions:** cInput is from scanf() cInput is 'Y', 'y', 'N', or 'n'	7	scanf() cInput = 'Y'	**char ***	cInput	scanf() => 'Y'	**int**	*function return*	1	**int**	*function return*	1	Pass
		scanf() cInput = 'y'	**char ***	cInput	scanf() => 'y'	**int**	*function return*	1	**int**	*function return*	1	Pass
		scanf() cInput = 'N'	**char ***	cInput	scanf() => 'N'	**int**	*function return*	0	**int**	*function return*	0	Pass
		scanf() cInput = 'n'	**char ***	cInput	scanf() => 'n'	**int**	*function return*	0	**int**	*function return*	0	Pass
		scanf() cInput = 'A'	**char ***	cInput	scanf() => 'A'	Invalid input for cInput			Invalid input for cInput			Pass
void promptEnemyEquipConfirm **Preconditions:** ***nEnemyHP**, ***nEnemyATK**, ***nEnemyDEF**, and ***nEnemySPD** are non-negative and less than or equal to 999 **Input conditions:** nWeaponInput and nArmorInput are from rand() % 4 nWeaponInput and nArmorInput are 0, 1, 2, or 3	8	rand() % 4 nWeaponInput = 0 nArmorInput = 0	**int ***	nEnemyHP	100	**int ***	nEnemyHP	100	**int ***	nEnemyHP	100	Pass
			int *	nEnemyATK	10	**int ***	nEnemyATK	25	**int ***	nEnemyATK	25	Pass
			int *	nEnemyDEF	10	**int ***	nEnemyDEF	13	**int ***	nEnemyDEF	13	Pass
			int *	nEnemySPD	10	**int ***	nEnemySPD	16	**int ***	nEnemySPD	16	Pass
			int *	nEnemyCrit	15	**int ***	nEnemyCrit	15	**int ***	nEnemyCrit	15	Pass
			int	nWeaponInput	rand() % 4 => 0							
			int	nArmorInput	rand() % 4 => 0							
		rand() % 4 nWeaponInput = 1 nArmorInput = 1	**int ***	nEnemyHP	100	**int ***	nEnemyHP	105	**int ***	nEnemyHP	105	Pass
			int *	nEnemyATK	10	**int ***	nEnemyATK	30	**int ***	nEnemyATK	30	Pass
			int *	nEnemyDEF	10	**int ***	nEnemyDEF	18	**int ***	nEnemyDEF	18	Pass
			int *	nEnemySPD	10	**int ***	nEnemySPD	10	**int ***	nEnemySPD	10	Pass
			int *	nEnemyCrit	15	**int ***	nEnemyCrit	15	**int ***	nEnemyCrit	15	Pass
			int	nWeaponInput	rand() % 4 => 1							
			int	nArmorInput	rand() % 4 => 1							
		rand() % 4 nWeaponInput = 2 nArmorInput = 3	**int ***	nEnemyHP	100	**int ***	nEnemyHP	100	**int ***	nEnemyHP	100	Pass
			int *	nEnemyATK	10	**int ***	nEnemyATK	50	**int ***	nEnemyATK	50	Pass
			int *	nEnemyDEF	10	**int ***	nEnemyDEF	10	**int ***	nEnemyDEF	10	Pass
			int *	nEnemySPD	10	**int ***	nEnemySPD	15	**int ***	nEnemySPD	15	Pass
			int *	nEnemyCrit	15	**int ***	nEnemyCrit	15	**int ***	nEnemyCrit	15	Pass
			int	nWeaponInput	rand() % 4 => 2							
			int	nArmorInput	rand() % 4 => 3							
void processAttack **Preconditions:** **nActorATK**, ***nTargetCurrentHP**, and **nTargetDEF** are non-negative and less than or equal to 999 **nActorCrit** is non-negative and less than or equal to 100 **nActorIsPlayer** is 0 or 1 **Input conditions:** nCritRoll is from rand() % 100 nCritRoll ranges from 0 to 99	9	nTargetDEF ≤ nActorATK nActorATK - nTargetDEF ≤ nTargetCurrentHP rand() % 100 nCritRoll ≥ nActorCrit	**int**	nActorATK	50	**int ***	nTargetCurrentHP	60	**int ***	nTargetCurrentHP	60	Pass
			int	nActorCrit	15							
			int *	nTargetCurrentHP	100							
			int	nTargetDEF	10							
			int	nActorIsPlayer	1							
			int	nCritRoll	rand() % 100 => 99							
		nTargetDEF > nActorATK nActorATK - nTargetDEF ≤ nTargetCurrentHP rand() % 100 nCritRoll ≥ nActorCrit	**int**	nActorATK	50	**int ***	nTargetCurrentHP	100	**int ***	nTargetCurrentHP	100	Pass
			int	nActorCrit	15							
			int *	nTargetCurrentHP	100							
			int	nTargetDEF	60							
			int	nActorIsPlayer	1							
			int	nCritRoll	rand() % 100 => 99							
		nActorATK ≤ nTargetCurrentHP rand() % 100 nCritRoll < nActorCrit	**int**	nActorATK	50	**int ***	nTargetCurrentHP	50	**int ***	nTargetCurrentHP	50	Pass
			int	nActorCrit	15							
			int *	nTargetCurrentHP	100							
			int	nTargetDEF	10							
			int	nActorIsPlayer	1							
			int	nCritRoll	rand() % 100 => 1							

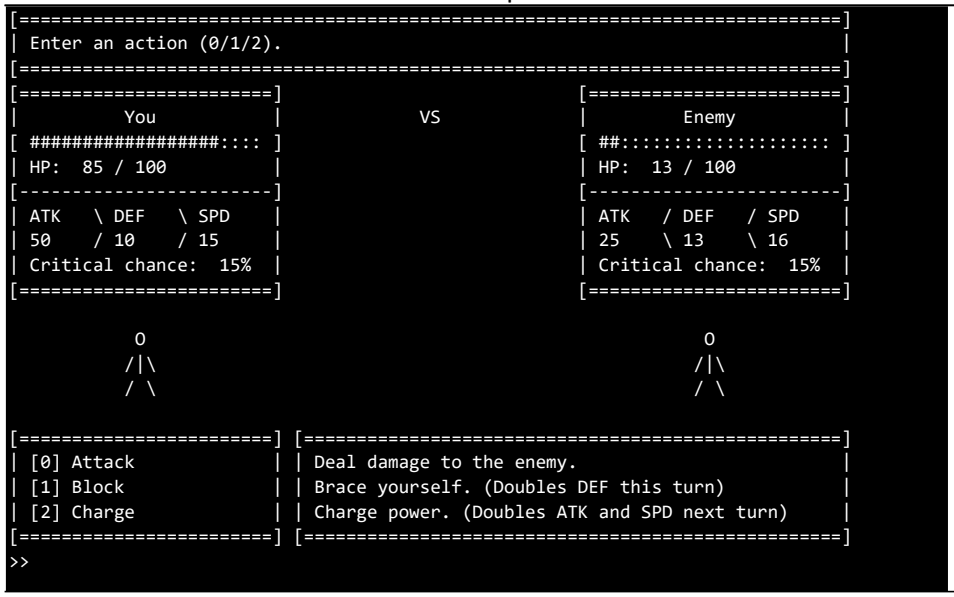
Function name	#	Test description	Sample input			Expected result			Actual result			Pass / fail
			Type	Name	Value / value at address	Type	Name	Value / value at address	Type	Parameter name	Value / value at address	
		nActorATK - nTargetDEF > nTargetCurrentHP rand() % 100 nCritRoll ≥ nActorCrit	int	nActorATK	999	int *	nTargetCurrentHP	0	int *	nTargetCurrentHP	0	Pass
			int	nActorCrit	15							
			int *	nTargetCurrentHP	100							
			int	nTargetDEF	10							
			int	nActorIsPlayer	1							
			int	nCritRoll	rand() % 100 => 99							
void processBlock **Preconditions:** ***nActorDEF** is non-negative and less than or equal to 999 **nActorIsPlayer** is 0 or 1	10	nActorDEF = 10	int *	nActorDEF	10	int *	nActorDEF	20	int *	nActorDEF	20	Pass
		nActorDEF = 50	int *	nActorDEF	50	int *	nActorDEF	100	int *	nActorDEF	100	Pass
		nActorDEF = 999	int *	nActorDEF	999	int *	nActorDEF	1998	int *	nActorDEF	1998	Pass
void processCharge **Preconditions:** nActorATK and nTargetSPD are non-negative and ≤ 999	11	nActorATK = 10 nActorSPD = 10 nActorIsCharging = 0	int *	nActorATK	10	int *	nActorATK	20	int *	nActorATK	20	Pass
			int *	nActorSPD	10	int *	nActorSPD	20	int *	nActorSPD	20	Pass
			int *	nActorIsCharging	0	int *	nActorIsCharging	0	int *	nActorIsCharging	0	Pass
			int	nActorIsPlayer	A							
		nActorIsCharging = 1	int *	nActorATK	20	int *	nActorATK	20	int *	nActorATK	20	Pass
			int *	nActorSPD	20	int *	nActorSPD	20	int *	nActorSPD	20	Pass
			int *	nActorIsCharging	1	int *	nActorIsCharging	0	int *	nActorIsCharging	0	Pass
			int	nActorIsPlayer	1							
		nActorATK = 999 nActorSPD = 999 nActorIsCharging = 0	int *	nActorATK	999	int *	nActorATK	1998	int *	nActorATK	1998	Pass
			int *	nActorSPD	999	int *	nActorSPD	1998	int *	nActorSPD	1998	Pass
			int *	nActorIsCharging	0	int *	nActorIsCharging	0	int *	nActorIsCharging	0	Pass
			int	nActorIsPlayer	1							

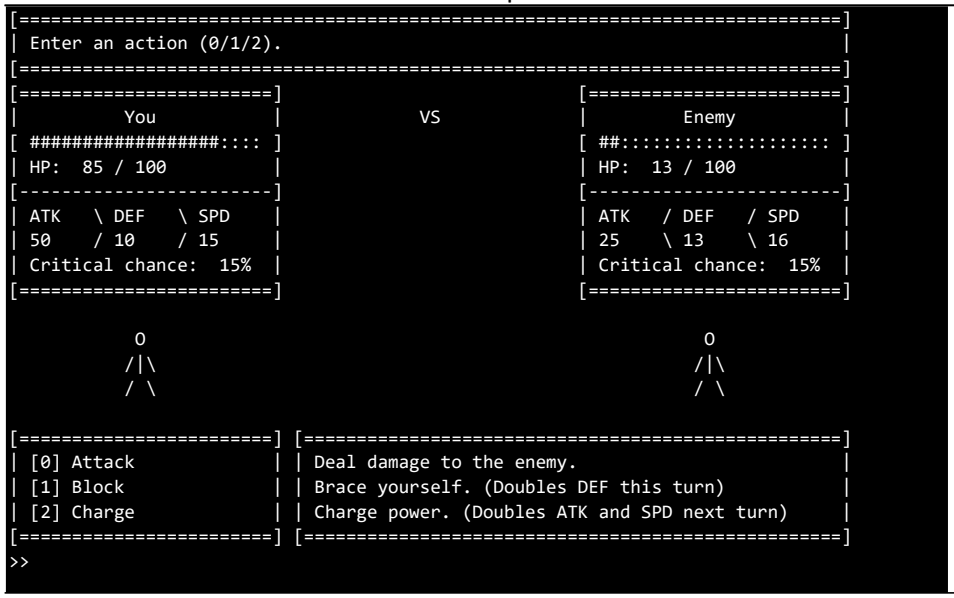
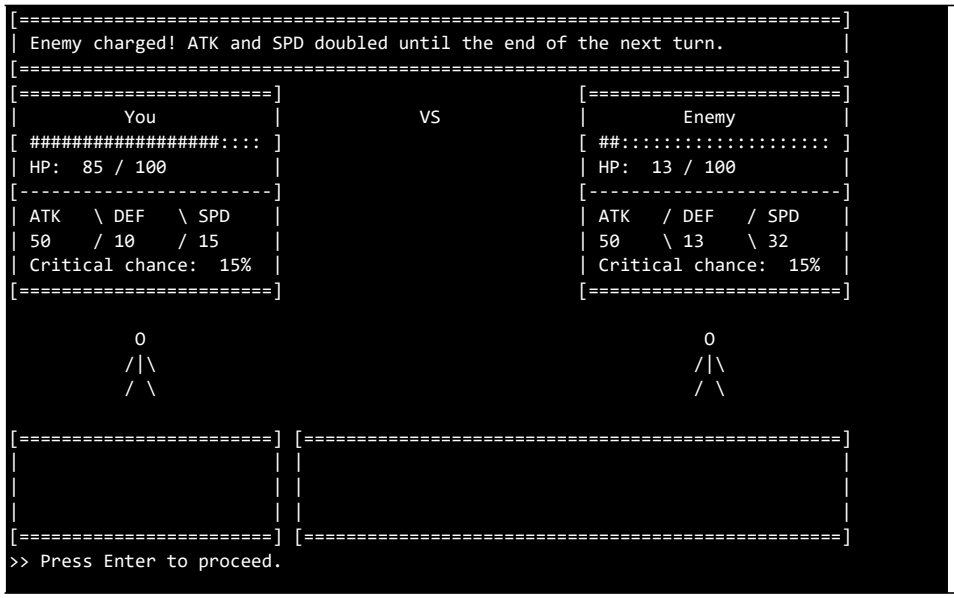
Screen Output	Event	Result
	User presses enter key	Program progresses
	User enters **0**	Program enters **Standard mode**
	User enters **2**	**Broadsword** is selected for player
	User presses enter key	Program progresses

Screen Output	Event	Result
[==========] | Select an armor. | [==========] [==========] [==========] | Armor | | Details | [----------] [----------] | [0] Mythril | | Lightweight armor that increases DEF and SPD. | | | (DEF + 3, SPD + 3) | [----------] [----------] | [1] Chainmail | | Standard chainmail armor that boosts DEF and HP | | | (DEF + 8, HP + 5) | [----------] [----------] | [2] Adamantite armor | | Heavy armor than boosts DEF and HP; lowers SPD. | | | (DEF + 18, HP + 10, SPD - 4) | [----------] [----------] | [3] No armor | | Wearing no armor greatly improves speed. | | | (SPD + 10) | [==========] [==========] [==========] [==========] | Choose. (0/1/2/3) | | Current stats: 100/10/10/10 | [==========] [==========] >>	User enters **3**	**No armor** is selected for player
[==========] | You selected the following armor: | | No armor | [==========] >> Press Enter to proceed.	User presses enter key	Program progresses
[==========] | You will battle with the following stats: | [==========] [==========] [==========] [==========] | Stats | | Weapon | | Armor | [----------] [----------] [----------] | ATK \ DEF \ SPD | | Broadsword | | No armor | | 50 / 10 / 15 | | | | | [----------] | | | | | | Base HP: 100 | | | | | | Critical chance: 15% | | | | | [==========] [==========] [==========] [==========] [==========] | Proceed? (Y/y/N/n) | | TIP: Canceling will let you change equipment. | [==========] [==========] >>	User enters **Y**	Program progresses
[==========] | The enemy is selecting their equipment! | [==========] >> Press Enter to proceed.	User presses enter key	Program progresses
[==========] | The enemy has chosen the following equipment: | [==========] [==========] [==========] [==========] | Stats | | Weapon | | Armor | [----------] [----------] [----------] | ATK \ DEF \ SPD | | Dagger | | Mythril | | 25 / 13 / 16 | | | | | [----------] | | | | | | Base HP: 100 | | | | | | Critical chance: 15% | | | | | [==========] [==========] [==========] >> Press Enter to proceed.	Program generates value for **rand() % 4** User presses enter key	**Dagger** and **Mythril** are selected for enemy Program progresses
[==========] | The enemy gladiator challenges you! | | Reduce their HP to 0 to win! | [==========] >> Press Enter to proceed.	User presses enter key	Program progresses

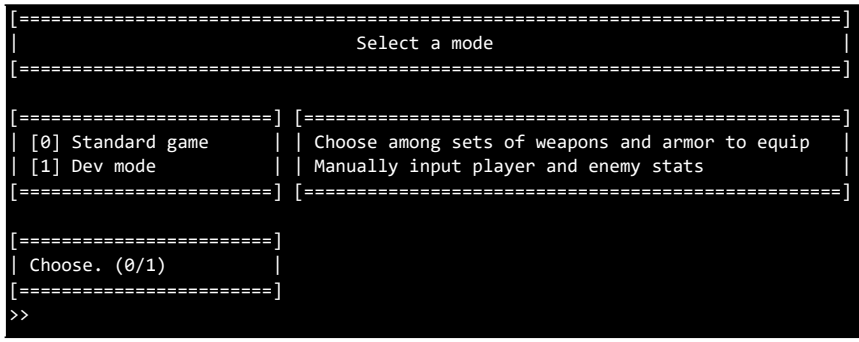
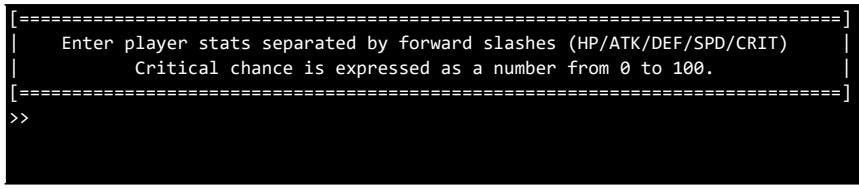
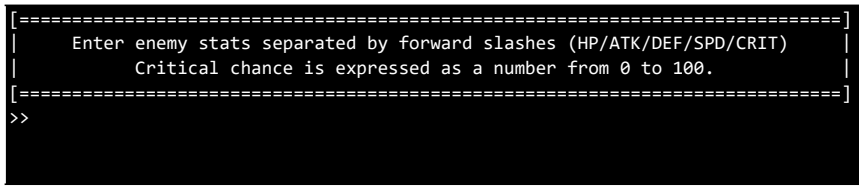
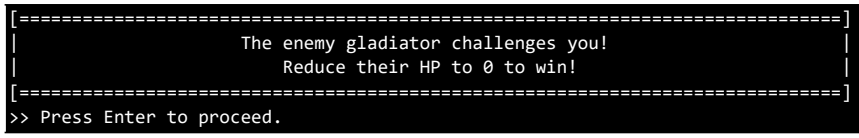
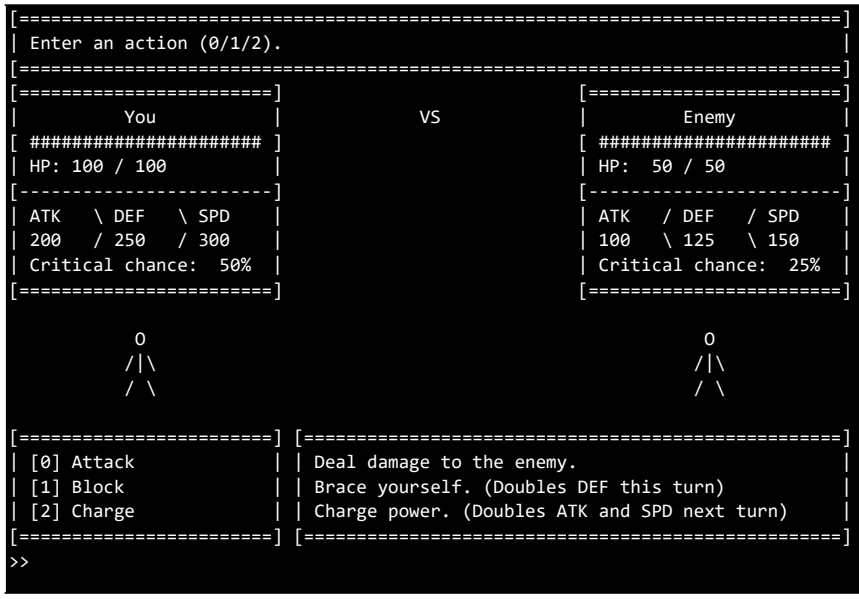
Screen Output	Event	Result
	User enters **2**	**Charge** action is selected for player
	Program generates value for **rand() % 3** Enemy executes **Block** action	**Block** action is selected for enemy Enemy **DEF** is doubled
	User presses enter key Player executes **Charge** action	Program progresses Player **ATK** and **SPD** are doubled
	User presses enter key	Program progresses

Screen Output	Event	Result
	Turn begins	Enemy **DEF** is reverted Player **charging flag** is enabled
	User enters 0	**Attack** action is selected for player
	Program generates value for **rand() % 3**	**Attack** action is selected for enemy
	Player executes **Attack** action	Enemy **HP** is reduced by Player **ATK** - Enemy **DEF** = **100** - **13** = **87**
	User presses enter key	Program progresses
	Enemy executes **Attack** action	Player **HP** is reduced by Enemy **ATK** - Player **DEF** = **25** - **10** = **15**
	User presses enter key	Program progresses

Screen Output	Event	Result
	Turn begins User enters 1 Program generates value for **rand() % 3**	Player **ATK** and **SPD** are reverted **Block** action is selected for player **Block** action is selected for enemy
	Player executes **Block** action User presses enter key	Player **DEF** is doubled Program progresses
	Enemy executes **Block** action User presses enter key	Enemy **DEF** is doubled Program progresses

Screen Output	Event	Result
	Turn begins	Player **DEF** is reverted Enemy **DEF** is reverted
	User enters 0	**Attack** action is selected for player
	Program generates value for **rand() % 3**	**Charge** action is selected for enemy
	Enemy executes **Charge** action	Enemy **ATK** and **SPD** are doubled
	User presses enter key	Program progresses
	Player executes **Attack** action	Enemy **HP** is reduced by Player **ATK** - Enemy **DEF** = **50** - **13** = **37** Since **37** > Enemy **HP**, it is instead reduced by Enemy **HP** = **13**
	User presses enter key	Program progresses
	User presses enter key	Program progresses

Screen Output	Event	Result
Program terminates		

Screen Output	Event	Result
	User presses enter key	Program progresses
	User enters **1**	Program enters **Dev mode**
	User enters **100/200/250/300/50**	Player is assigned the following stats: **HP: 100** **ATK: 200** **DEF: 250** **SPD: 300** **Critical Chance: 50%**
	User enters **50/100/125/150/25**	Enemy is assigned the following stats: **HP: 50** **ATK: 100** **DEF: 125** **SPD: 150** **Critical Chance: 25%**
	User presses enter key	Program progresses
		

End of test