CAASI, SAMANTHA NICOLE L.

S16A

int getInt()					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Integer	nInput = 1	return nInput	returned nInput	P
2		nInput = 5	return nInput	returned nInput	P
3		nInput = -3	return nInput	retured nInput	P

int promptOneZero()					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Go back to main menu	nChoice = 1	returns nChoice	returned nChoice	P
2	Exit	nChoice = 0	returns nChoice	returned nChoice	P
3	Invalid input	nChoice = -1	Loops until nChoice is either 1 or 2	Looped until nChoice is either 1 or 2	P

int promptInt(int nMin, int nMax)					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Valid input	nMin = 0 nMax = 6 nChoice = 1	return nChoice	returned nChoice	P
2	Invalid input (above range)	nMin = 1 nMax = 4 nChoice = 5	Loops until nChoice is within range	Looped until nChoice is within range	P
3	Invalid input (below range)	nMin = 1 nMax = 6 nChoice = -1	Loops until nChoice is within range	Looped until nChoice is within range	P

char *Capitalize(char *str)					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Convert name to uppercase	str = "sam"	str = "SAM"	str = "SAM"	P
2	Convert guess to uppercase	str = "poles"	str = "POLES"	str = "POLES"	P
3	Convert mixed word to uppercase	str = "blAdE"	str = "BLADE"	str = "BLADE"	P

void askName(char *name)					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Valid name	name = "S16"	changes the value of name from the function its called from	changes the value of name from the function its called from	P
2	Invalid name (exceeds 3 characters)	name = "asdfa"	Loops until name is within range	Loops until name is within range	P

3	Invalid name (less than 3 characters)	name = "po"	Loops until name is within range	Loops until name is within range	P

void saveRecord(int win, char name[], int timeP0, int nCopyTries, int nCWS, int *nNoPlayers, struct Player P[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Player exists and won	win = 1 nI = 1	call updateExisting(nI, timeP0, nCTries, P)	called updateExisting(nI, timeP0, nCTries, P)	P
2	Player exists and does not win	win = 0 nI = 3	nCTries = 0 call updateExisting(nI, timeP0, nCTries, P)	nCTries = 0 called updateExisting(nI, timeP0, nCTries, P)	P
3	Player does not exist and wins	win = 1 nI = -1 nNoPlayers = 3	call addNew(name, timeP0, nCTries, nCWS, nNoPlayers, P)	called addNew(name, timeP0, nCTries, nCWS, nNoPlayers, P)	P

void addNew(char name[], int timeP0, int nCopyTries, int nCWS, int *nNoPlayers, struct Player P[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Adding new player and respective initial data to the player.txt file	name = "sam" timeP0 = 5 nCTries = 3 nCWS = 1 nNoPlayers = 3 (nI = 3)	P[3].name = "sam" P[3].timePC = 5 P[3].timePB = 5 P[3].nCWS = 1 P[3].nHWS = 1 P[3].turns[2] = 1 nNoPlayers = 4	P[3].name = "sam" P[3].timePC = 5 P[3].timePB = 5 P[3].nCWS = 1 P[3].nHWS = 1 P[3].turns[2] = 1 nNoPlayers = 4	P
2		name = "ela" timeP0 = 10 nCTries = 1 nCWS = 4 nNoPlayers = 6 (nI = 6)	P[6].name = "ela" P[6].timePC = 10 P[6].timePB = 10 P[6].nCWS = 4 P[6].nHWS = 4 P[6].turns[3] = 1 nNoPlayers = 7	P[6].name = "ela" P[6].timePC = 10 P[6].timePB = 10 P[6].nCWS = 4 P[6].nHWS = 4 P[6].turns[3] = 1 nNoPlayers = 7	P
3		name = "rak" timeP0 = 4 nCTries = 6 nCWS = 2 nNoPlayers = 8 (nI = 8)	P[8].name = "rak" P[8].timePC = 4 P[8].timePB = 4 P[8].nCWS = 1 P[8].nHWS = 1 P[8].turns[1] = 1 nNoPlayers = 9	P[8].name = "rak" P[8].timePC = 4 P[8].timePB = 4 P[8].nCWS = 1 P[8].nHWS = 1 P[8].turns[1] = 1 nNoPlayers = 9	P

void updateExisting(int nI, int timeP0, int nCopyTries, struct Player P[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F

1	Current time is less than best time	nI = 1 timeP0 = 6 nCTries = 5 P[1]timePB = 9	increment P[1].nCWS by 1 update P[1].timePB to 6 increment P[1].turns[4] by 1	P[1].nCWS is incremented by 1 P[1].timePB = 6 P[1].turns[4] is incremented by 1	P
2	Current winstreak is higher than best winstreak	nI = 4 nCTries = 4 P[4].nCWS = 5 P[4].nHWS = 4	increment P[4].nCWS by 1 increment P[4].turns[3] by 1 update P[4].nHWS to 5	P[4].nCWS is incremented by 1 P[4].turns[3] is incremented by 1 P[4].nHWS = 5	P
3	Player did not win	nI = 7 nCTries = 0 timeP0 = 3	P[7].nCWS = 0 P[7].timePC = 3 other data should not be altered	P[7].nCWS = 0 P[7].timePC = 3 other data are not altered	P

`int searchPlayer(char name[], struct Player P[])`

#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Player does not exist	name = "sik" sik is not found in name member of Player P[]	returns -1	returned -1	P
2	Player code of letters exists	name = "sam" found P[1].name = "sam"	returns 1	returned 1	P
3	Player code of numbers exists	name = "143" found P[8].name = "143"	returns 8	returned 8	P

`void Wordle(char cGuess[], char Word[])`

#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Misplaced letters	cWord = "SLOPS" cGuess = "SPILL"	cClues = {S,+,x,+,x} prints "S+x+x"	printed "S+x+x"	P
2	No matching letters	cWord = "SPILL" cGuess = "AARGH"	cClues = {x,x ,x,x,x} prints "xxxxx"	printed "xxxxx"	P
3	Correct placement, almost correct	cWord = "POLES" cGuess = "MOLES"	cClues = {x,O,L,E,S} prints "xOLES"	printed "xOLES"	P

`void pickWord(char *cWord, char cFile[])`

#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Updates cWord (assuming default "dict.txt")	nRandom = 1686 cFile = "dict.txt"	updates cWord = "fermi"	updated cWord = "fermi"	P
2		nRandom = 2022 cFile = "dict.txt"	updates cWord = "giver"	updated cWord = "giver"	P
3		nRandom = 1921 cFile = "dict.txt"	updates cWord = "fussy"	updated cWord = "fussy"	P

void askSecret (char *cWord, char cFile[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	cWord entered is not on cFile	cWord = "adfaq"	prints "The word is not in the dictionary" loops until valid cWord is entered	prints "The word is not in the dictionary" looped until valid cWord is entered	P
2	cWord entered is on cFile	cWord = "aargh"	updates cWord = "aargh"	updated cWord = "aargh"	P
3	cWord is a string of integers	cWord = "12316"	prints "The word is not in the dictionary" loops until valid cWord is entered	prints "The word is not in the dictionary" looped until valid cWord is entered	P

void askGuess (char *cGuess, char cFile[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	cWord entered is not on cFile	cGuess = "zeqty"	prints "The word is not in the dictionary" loops until valid cGuess is entered	prints "The word is not in the dictionary" looped until valid cGuess is entered	P
2	cWord entered is on cFile	cGuess = "zowie"	updates cGuess = "zowie"	updated cGuess = "zowie"	P
3	cWord is a string of integers and letters	cGuess = "121va"	prints "The word is not in the dictionary" loops until valid cGuess is entered	prints "The word is not in the dictionary" looped until valid cGuess is entered	P

int getWordCount(char cFile[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Dictionary file contains nothing	dict.txt with 0 words	return 0	returned 0	P
2	Dictionary file contains words below the maximum limit	dict.txt containing 5155 words	return 5155	returned 5155	P
3	Dictionary file exceeds maximum limit of entries	dict.txt containing 6004 words	return 6000	returned 6000	P

int checkDict(char cCheck[], char cFile[])					
#	Description	Sample Input	Expected Result	Actual Result	P/F
1	Word is not on the dictionary file	cCheck = "hatdg" hold = "HATDG"	return 1	returned 1	P
2	Word is on the dictionary file	cCheck = "poles" hold = "POLES"	return 0	returned 0	P
3	Word is on the dictionary file	cCheck = "blade" hold = "BLADE"	return 1	returned 1	P