CSE 102 –ComputerProgramming
HW 08

Last Submission Date: May 9, 2018 –23:30

Part 1-

91	92	T	94	95	96	97	S {72}	S {56}	100 (Finish)
90	M {94}	88	87	86	P	84	83	B	81
71	72	S {68}	74	75	M {83}	77	78	T	80
70	M {77}	68	67	B	65	64	S {52}	62	61
51	52	53	T	55	56	M {77}	58	59	S {44}
B	49	P	47	46	S {22}	44	43	42	41
31	M {52}	33	34	35	B	37	38	39	T
30	B	28	27	26	S {12}	24	23	22	21
11	12	P	14	15	16	M {28}	18	19	20
10	9	8	M {11}	S {1}	5	4	3	2	1 (Start)

Fig-1 The 100 games

Purpose of the 100 game: Each player rolls one dice by turns and try to reach 100 block on the board as soon as possible. The player win the game who reaches 100 block firstly.

Some rules of the game:

- Each player can roll the dice just once at his/ her turn and move forward by on the number of the dice.
- When a player reaches the block 100, the game is over.
- After the dice rolling, if the move overshoots the block 100, then the player does not move.
- Blocks with label “P” penalties. Similar to the first rule, if a player ends up in a “P” block after dice rolling, then the player does not move.
- Blocks with label “S” represents snakes. If the player reaches an “S” block, returns the block that is shown in brackets.
- Blocks with label “M” represents stairs. If the player reaches an “M” block, jumps the block that is shown in brackets.
- Blocks with label “B” represents boosts. If the player reaches a “B” block, jumps from current block to fifth next block.
- Blocks with label “T” represents traps. If the player reaches a “T” block, returns from current block to fifth previous block.

a) Write the code of the function that initialize the board that is shown in fig-1. The board prototype should be like ;
struct block board[10][10]

b) Write the code of the function that prints the board like shown in fig-1. Double row cells can be printed plain (like M {11}). Don’t draw the lines which covers the blocks.

c) Write the code of the function that plays the game as a single player, recursively. The function tries to reach 100. When the function reaches “100” block, it prints the moves from last to first and returns move counts.

d) Write the code of the function that plays two separate game that one game for player 1 and another for player 2. Compare which player won the game and print it.

Notes:

- Define a **block** named structure for blocks. The structure must have **text, data, type, pos_x, pos_y, jump_x, jump_y.** For example;

  text= M {11}
  data= 7
  type=M
  pos_x=9
  pos_y=3
  ~~block= block 11 (just represent, not the code)~~
  jump_x=8
  jump_y=0
  You can add another properties to the structure if you needed.

- Define an enumerated data type for block types.

Part 2-

Write the following recursive function:

char* find_operations(int arr[], int expected_val, char operations[], size_t arr_len, int cursor, int current)

1. The first argument is an array of numbers. Assume that n numbers are given.
2. The second argument is an integer number.
3. The third argument is a string that has n-1 spaces at first call.
4. Your function finds a sequence of operators (exactly n-1 of those) such that successive application of these operators on the given array of numbers produces the given number (the second argument). The operators can be one of {+, -, *}.

For example, for array {25, 12, 6, 10, 32, 8}, the operators {-, *, -, -, +} produce 44 by applying (((((25 - 12) * 6) - 10) - 32) +8).

Applying the operators in sequence means that the first operator is applied to the first two numbers in the array. Assume that the result is a12. Then the second operator is applied to a12 and the third number in the array. Assume that this in turn is labeled as a123. Then the third operator will be applied to a123 and the fourth number in the array. This will continue till all the operators are applied to the numbers in the array sequentially.

There may be multiple answers for the sequence of operators. You are expected to find one that works. In some cases, the given the argument may not have any n-1 operators.
For example {1, 2, 3} and 100 does not have any solution. In this case, your function should return a list of spaces (n-1 of them).

Notes:

1. Do not change the given function prototype. But you are allowed to use auxiliary functions as you see fit. However, your implementation of the main algorithm should be recursive. Non recursive implementations will not be graded and will receive a 0 for this part.
2. You are not allowed to use any global and static variables in your function implementations. Any use of such variables will result in a 0 for this part.

General:

1. Obey honor code principles.
2. **Read your homework carefully** and follow the directives about the I/O format (data file names, file formats, etc.) and submission format **strictly**. Violating any of these directives will be penalized.
3. Obey coding convention.
4. Your submission is HW08_studentnumber.zip and include the following files and NOTHING MORE (no data files, object files, etc):

- HW08part1.c
- HW08part2.c

5. Do not use non-English characters in any part of your homework (in body, file name, etc.).
6. Deliver the print out of your work until the last submission date.
7. For questions about homework, you can send an email to b.koca@gtu.com.tr