2022 Fall Data Structure HW3

Handwriting

1. (40%) Using manual transformation, write the following infix expressions in their postfix and prefix forms:

   a. D – B + C
   b. A * B + C * D
   c. (A + B) * C – D * F + C
   d. (A – 2 * (B + C) – D * E) * F

2. (20%) If the values of A, B, C, and D are 2, 3, 4, and 5, respectively, manually calculate the value of the following prefix expressions:

   a. + – * A B C D
   b. – * A + B C D

3.(40%) Change the following infix expressions to postfix expressions using the algorithmic method (a stack):

   c. (A + B) * C – D * F + C
   d. (A – 2 * (B + C) – D * E) * F

*請附上想法 / 計算過程，只寫答案會斟酌扣分。
寫法可參考下方格式或是課本範例，好理解即可。

Original	Stack	Output
A * B + C * D		
* B + C * D		A
B + C * D	*	A
+ C * D	*	A B
C * D	+	A B *
* D	+	A B * C
D	* +	A B * C
	* +	A B * C D
		A B * C D * +

Programming

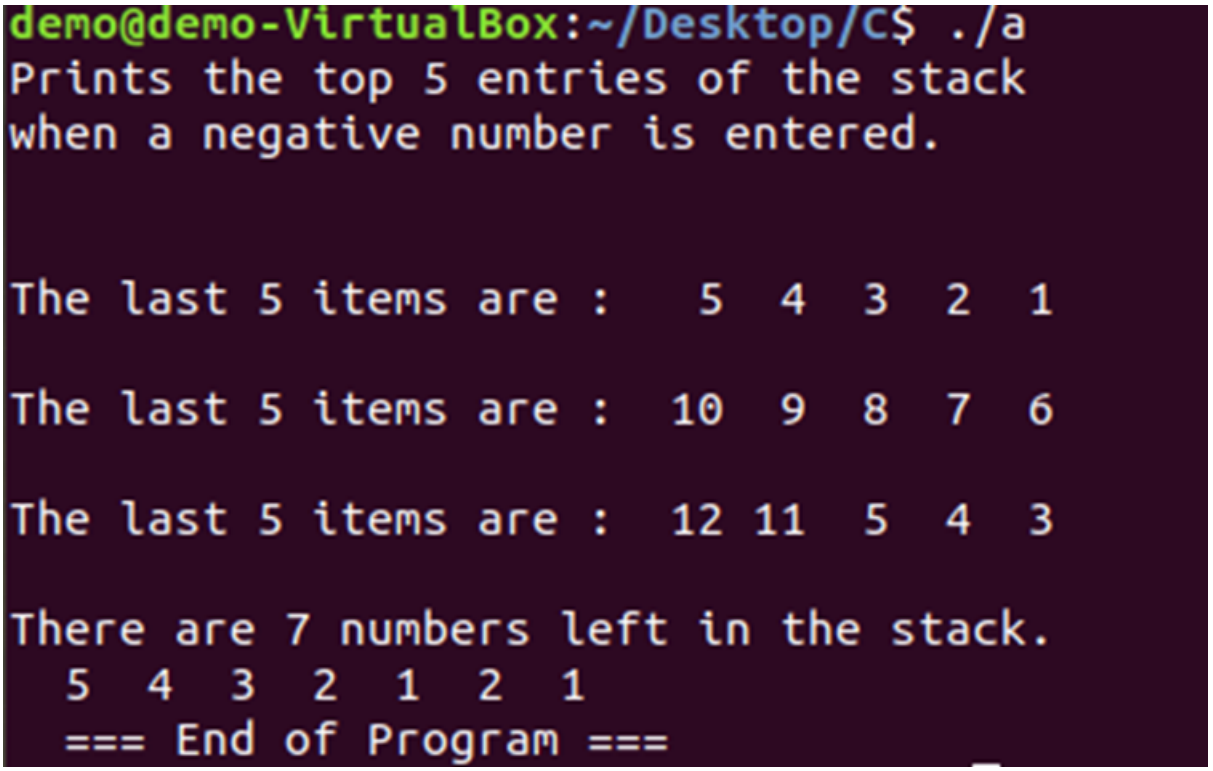
4. (50%)

One of the applications of a stack is to backtrack—that is, to retrace its steps. As an example, imagine we want to read a list of items, and each time we read a negative number we must backtrack and print the five numbers that come before the negative number and then discard the negative number.

Use a stack to solve this problem. Read the numbers in input.txt (it is in e3) and push them into the stack (without printing them) until a negative number is read. At this time, stop reading and pop five items from the stack and print them out. If there are fewer than five items in the stack, print an error message(error message is “ERROR”) and stop the program.

After printing the five items, resume reading data in input.txt and placing them in the stack. When the end of the file is detected, print a message and the items remaining in the stack.

In this question we need to use functions such as createStack, popStack, pushStack to meet the requirements for this question. These three functions are written in the textbook (Stack ADT section). We need to read the numbers in input.txt to verify the result of your program. The result should be as follows.



5. (50%)

24. Write a program that simulates a mouse in a maze. The program must print the path taken by the mouse from the starting point to the final point, including all spots that have been visited and backtracked. Thus, if a spot is visited two times, it must be printed two times; if it is visited three times, it must be printed three times.

The maze is shown in Figure 3-25. The entrance spot, where the mouse starts its journey, is chosen by the user who runs the program. It can be changed each time.

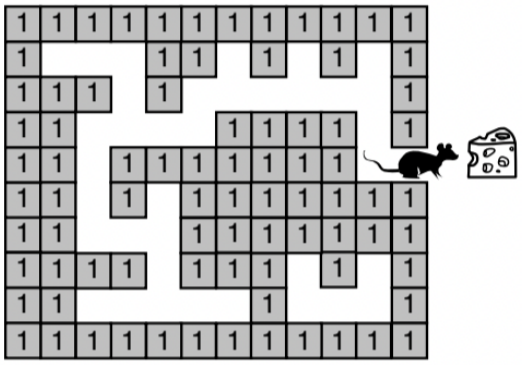


FIGURE 3-25 Mouse Maze for Project 24

A two-dimensional array can be used as a supporting data structure to store the maze. Each element of the array can be black or white. A black element is a square that the mouse cannot enter. A white element is a square that can be used by the mouse. In the array a black element can be represented by a 1 and a white element by a 0.

When the mouse is traversing the maze, it visits the elements one by one. In other words, the mouse does not consider the maze as an array of elements; at each moment of its journey, it is only in one element. Let's call this element the `currentSpot`. It can be represented by a structure of two integer fields. The first field is the row and the second is the column coordinate of the spot in the maze. For example, the exit in Figure 3-25 is at (5,12)—that is, row 5, column 12.

The program begins by creating the maze. It then initializes the exit spot and prompts the user for the coordinates of the entrance spot. The program must be robust. If the user enters coordinates of a black spot, the program must request new coordinates until a white spot is entered. The mouse starts from the entrance spot and tries to reach the exit spot and its reward. Note, however, that some start positions do not lead to the exit.

As the mouse progresses through its journey, print its path. As it enters a spot, the program determines the class of that spot. The class of a spot can be one of the following:

a. Continuing—A spot is a continuing spot if one and only one of the neighbors (excluding the last spot) is a white spot. In other words, the mouse has only one choice.

b. Intersection—A spot is an intersection spot if two or more of the neighbors (excluding the last spot) is a white spot. In other words, the mouse has two or more choices.
c. Dead end—A spot is a dead-end spot if none of the neighbors (excluding the last spot) is a white spot. In other words, the mouse has no spot to choose. It must backtrack.
d. Exit—A spot is an exit spot if the mouse can get out of the maze. When the mouse finds an exit, it is free and receives a piece of cheese for a reward.

To solve this problem, you need two stacks. The first stack, the visited stack, contains the path the mouse is following. Whenever the mouse arrives at a spot, it first checks to see whether it is an exit. If not, its location is placed in the stack. This stack is used if the mouse hits a dead end and must backtrack. Whenever the mouse backtracks to the last decision point, also print the backtrack path.

When the mouse enters an intersection, the alternatives are placed in a second stack. This decision point is also marked by a special decision token that is placed in the visited stack. The decision token has coordinates of (–1,–1). To select a path, an alternative is then popped from the alternatives stack and the mouse continues on its path.

While backtracking, if the mouse hits a decision token, the token is discarded and the next alternative is selected from the alternatives stack. At this point print an asterisk (*) next to the location to show that the next alternative path is being selected.

- 題目的座標敘述是以(1, 1)作為二維陣列的起點，但本次作業請同學以0為x軸、y軸起始座標，也就是說(0, 0)此點為陣列的起始座標，可參考以下範例說明。
- 本題的方位順序依序為：North>>South>>West>>East，請同學依此作為設計stack順序的標準。
- 不用紀錄每個位置經過幾次，僅需將鼠標經過的位置以2作為輸出。
- Input: 請讀取[input.txt]作為初始參數，1為不能進入的區域，0為可進入的區域
- Output: 請將Maze包含鼠標走過的路徑輸出為[answer.txt]，每個數字之間請以’ ’(空白符)區隔。
- 程式檔與input.txt、answer.txt在同一目錄下執行，僅需繳交程式檔。

[input.txt]
第一行：Maze維度
第二行：Starting Point
第三行：Final Point
其餘行：Maze資訊

input.txt

```
10 12
1 1
8 10
1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 1 1 1 1 1 1 1
1 1 1 0 1 1 0 0 0 0 1 1
1 1 1 0 1 1 0 1 1 0 1 1
1 1 1 0 0 0 0 1 1 0 1 1
1 1 1 0 1 1 0 1 1 0 1 1
1 1 1 0 1 1 0 1 1 0 1 1
1 1 1 1 1 1 0 1 1 0 1 1
1 1 0 0 0 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1
```

[answer.txt]

answer.txt

```
1 1 1 1 1 1 1 1 1 1 1 1
1 2 2 2 1 1 1 1 1 1 1 1
1 1 1 2 1 1 2 2 2 2 1 1
1 1 1 2 1 1 2 1 1 2 1 1
1 1 1 2 2 2 2 1 1 2 1 1
1 1 1 2 1 1 0 1 1 2 1 1
1 1 1 2 1 1 0 1 1 2 1 1
1 1 1 1 1 1 0 1 1 2 1 1
1 1 0 0 0 0 0 0 1 2 2 1
1 1 1 1 1 1 1 1 1 1 1 1
```

Submission - Deadline: 2022/10/25 13:20

題目形式:

- 手寫題可以用手寫拍照、打字的方式完成，但最後要統一轉成.pdf檔繳交
  檔名為HW3_{學號}.pdf。例如:HW3_0123456.pdf
- 程式題則繳交程式原始碼(.c檔/.cpp檔)
  檔名為HW3_{題號}_{學號}.c / .cpp。例如:HW3_4_0123456.c / .cpp

繳交方式:

- 將上述總共三個檔案(手寫題pdf檔*1+程式題c/cpp檔*2)直接上傳至e3
- 檔名 / 格式錯誤者扣該次作業總分10分。
- 程式部分輸出格式請照作業說明，若不同會酌量扣分。

收作業規則:

- 遲交一個禮拜內分數打七折，超過一個禮拜即不接受補交。
- 遲交期限內僅接受原本沒交作業的同學補交，不接受先前交過作業的同學再次補交，
  若要修改答案請在繳交期限內修改完畢。
- 請務必重新整理，確認檔案已成功上傳至e3。

如有任何問題，麻煩從e3來信給所有助教。