1. (1) Modify the parenthesis matching program to include two types of parentheses () and [] in expressions. Execute your program by keying in (((a+b)]*[[[c+d)]. The output will be

(2,6)
right parenthesis] at 7 has no matching left parenthesis [
right parenthesis) at 15 has no matching left parenthesis (
[10,16]
left parenthesis [at 9 has no matching right parenthesis]
left parenthesis (at 0 has no matching right parenthesis)

(2) Modify your program in Problem 1(1) to let the output be as follows if you key in the same input.

(2,6)
right parenthesis] at 7 has no matching left parenthesis [
right parenthesis) at 15 has no matching left parenthesis (
[11,16]
left parenthesis [at 10 has no matching right parenthesis]
left parenthesis [at 9 has no matching right parenthesis]
left parenthesis (at 1 has no matching right parenthesis)
left parenthesis (at 0 has no matching right parenthesis)

(3) Modify the parenthesis matching program to include 3 types of parentheses {}, [] and () in expressions. Execute your program by keying in {(a+b)*c]}}/{[d*[e+f]. The output will be

(1,5)
right parenthesis] at 8 has no matching left parenthesis [
{0,9}
right parenthesis } at 10 has no matching left parenthesis {
[16,20]
left parenthesis [at 13 has no matching right parenthesis]
left parenthesis { at 12 has no matching right parenthesis }

Take a screenshots of values of symb, pos, top, and stack[top].type and stack[top].pos **after each push and each pop** in the program using debug (F5), break points, add watch, step (F7) of Dev-C++. **Draw figures** to show the stack change (by showing **Push ?**, **Pop out ?** and **New stack contents**) and pop-related program output at each screenshot.

2. **Write a program** in the following **continuous** steps

(1) Use **rand()%100+1** to get 18 random numbers, output the numbers (one by one, one space in between, and **9 numbers in one line**) and

push the numbers into a created stack (**struct stack S)** one by one.

(2) Assign and output integer i the bottom element from the top of **S**, leaving **S** unchanged.

(3) Assign and output integer j the 2nd element from the bottom of **S**, leaving **S** unchanged.

(4) Assign and output integer k the 3rd element from the bottom of **S,** leaving **S** unchanged.

(5) Output the numbers from the top to the bottom of **S** (one by one, one space in between, and **9 numbers in one line),** leaving **S** unchanged.

(6) Assign and output integer m the 4th element from the bottom of **S**.

3. (1) Assume a queue is implemented by a **circular array**. **Comment in front of the program in (2)** the conditions respectively for an empty queue and a full queue if

(a) **front** points to two position before the queue head and **rear** points to two position after the queue tail

(b) **front** points to two position after the queue head and **rear** points to two position before the queue tail.

(2) **Write a program** in the following **continuous** steps (using **lastOperationIsDeleteq**).

(a) Use **rand()%100+1** to get 18 random numbers, output the numbers (one by one, one space in between, and **9 numbers in one line**) and add the numbers into a created queue (**struct queue Q**) one by one

(b) Assign and output integer n the 10th element from the head of **Q**, leaving **Q** unchanged.

(c) Output the numbers from the head to the tail of **Q** (one by one, one space in between, and **9 numbers in one line**).

(3) **Write a program** in the following **continuous** steps (using **TotalinQueue**).

(a) Use **rand()%100+1** to get 18 random numbers, output the numbers (one by one, one space in between, and **9 numbers in one line**) and add the numbers into a created queue (**struct queue Q**) one by one.

(b) Assign and output integer t the 4th element from the tail of **Q**.

(c) Output the numbers from the head to the tail of **Q** (one by one, one space in between, and **9 numbers in one line**).

(4) **Write a program** in the following **continuous** steps (using

sacrificing an element space).

(a) Use **rand()%100+1** to get 18 random numbers, output the numbers (one by one, one space in between, and **9 numbers in one line**) and add the numbers into a created queue (**struct queue Q**) one by one.

(b) Assign and output integer x the 12th element from the head of **Q**.

(c) Output the numbers from the head to the tail of **Q** (one by one, one space in between, and **9 numbers in one line**), leaving **Q** unchanged.

(d) Assign and output integer y the 8th element from the head of **Q**.

Note: Please

(1) put necessary **English Dev-C++ DEBUG window screenshots** to show required **Dev-C programs** and **highlighted required execution results**,

(2) comment student ID+your name **in every screenshots**, and

(3) put reports into one word file named by student_ID+your_name.