1. **Write a program** including the following steps (using **Linked Stacks**).

(1) Use **rand()%100+1** to get 30 random numbers, output the numbers (one by one, one space in between, and **8 numbers in one line**) and push the numbers into **S** one by one.

(2) Assign and output integer x the 11th element from the top of **S**, leaving **S** unchanged.

(3) Put integer x in (2) under the bottom of **S** (leaving the rest of **S** unchanged) and output the numbers (one by one, one space in between, and **8 numbers in one line**) on **S** from the top to the bottom.

(4) Execute the modified parenthesis matching program(**using the Linked Stack** created in this problem) by keying in [(((a+b)]\*[[[c+d)]. The output will be

(3,7)

right parenthesis ] at 8 has no matching left parenthesis [

right parenthesis ) at 16 has no matching left parenthesis (

[11,17]

left parenthesis [ at 10 has no matching right parenthesis ]

left parenthesis ( at 1 has no matching right parenthesis )

left parenthesis [ at 0 has no matching right parenthesis ]

2. (1) **Write a program** including the following steps (**implementing Linked Queues by Linked Stacks in Problem 1**).

(a) Use **rand()%100+1** to get 30 random numbers, output the numbers (one by one, one space in between, and **8 numbers in one line**) and add the numbers into **Q** one by one.

(b) Assign and output integer j the 8th element from the head of **Q**, leaving **Q** unchanged.

(c) Assign and output integer m the 3rd element from the tail of **Q**, leaving **Q** unchanged.

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

(2) **Write a program** including the following steps (**implementing Linked Stacks by Linked Queues**). **(Note: Be sure to assume the stack top is the queue front.)**

(a) Use **rand()%100+1** to get 30 random numbers, output the numbers (one by one, one space in between, and **8 numbers in one line**) and push the numbers into a created stack **S** one by one.

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

(c) Assign and output integer i the 12th element from the top of **S**.

3. **Write a program** including the following steps.

(1) Use **rand()%100+1** to get 50 random numbers, **output** the numbers (one by one, one space in between, and **8 numbers in one line**), **insert** the odd numbers into a created **linear** list (**listPointer p0**) one by one (sorted in ascending order – from small to big), and **insert** the even numbers into another created **linear** list (**listPointer p1**) one by one (sorted in ascending order – from small to big). (**Hint:** use the function void insertnum(listPointer \*p, int x).)

(2) output the linear list data from **p0** and **p1** (one by one from the first to the last, one space in between, and **8 numbers in one line**). (**Hint:** use the function void printL(listPointer p). )

(3) produce a new linear list (**listPointer p**) that contains the linear list **p0** followed by the linear list **p1**, and output the linear list data from the new linear list (**listPointer p**)(one by one from the first to the last, one space in between, and **8 numbers in one line**). (**Hint:** use the function listPointer concatenate(listPointer list1, listPointer list2). )

(4) change the new linear list **p** to a **circular** list. (**Hint:** use the function listPointer Lin2Cir(listPointer p). )

(5) output the circular list data(one by one from the first to the last, one space in between, and **8 numbers in one line**). (**Hint:** use the function void printC(listPointer p). )

(6) start from the first node of the circular list in (5), count around the list and delete the 3rd node each time (using int delete(listPointer p)), and output the deleted number until the list becomes empty.

4. **Write a program** including the following steps (about **array implementation of trees**).

(1) Use **rand()%100+1** to get 50 random numbers, **output** the numbers (one by one, one space in between), **insert** the numbers into a created **array** one by one (from tree[0]~tree[49]) immediately to form a tree implemented by an array.

(2) output the data in the tree nodes one by one in *preorder*.

(3) output the data in the tree nodes one by one in *inorder*.

(4) output the data in the tree nodes one by one in *postorder*.

(5) output the data in the tree nodes one by one in *levelorder*.

(6) output the data in the tree nodes one by one in *inorder* (using iterative inorder traversal).

**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.