Q1. Find time complexity for the following two codes. Specify best case and worst case, if exist (compute time for both of them). You are not required to give asymptotic complexity, just write time expression: **(4+5+3+3=15)**

```
a. i = 1;
    j = N;
    while (j > 1){
       cout << i << ' ';
      i++;
      if (i == N){
         i = 1;
         j = j – 1;
      }
    }
```

```
b.
i = 1
while i <= N
   if (x[i] > 50)
      k = 1
      while k <= N
         k = k * 2
      end-while
   }
   i = i + 1
}
```

c. Write asymptotic time complexity for each of the following:

i. **$2n\log(n) + 10000(\log(n))^{10} + 500n + 3.5n^2 + 2n^2$**: _____

ii. **$2n + 50000\log n$**: _____

iii. **$2n + 100900$:** _____

d. For the code below, you need to identify the exact number of times statements **S1** and **S2** executes: **(3 Marks)**

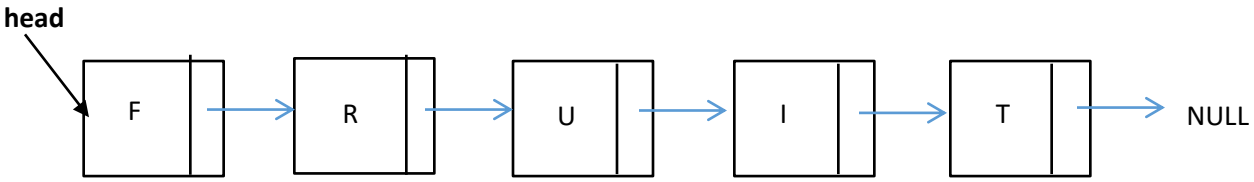
```
sum = 0;
for (i = 1; i <= n-1; i++)          ..... S1
    for (j = 1; j < i ; j += 2)
        sum++                       ..... S2
```

S1: _____

S2: _____

Q.2 Given the following function and a Linked List populated with specific nodes, as shown in the figure. Each node in the Linked List contains a data item of type **char** and has a pointer named **next,** which points to the next node. What would be the output of the function named **recursionP,** if called with head node? **(3 Marks)**

Linked List:




```
void recursionP(Node* head){
      if(!head)    return;
      if (head -> next)
            cout <<  head -> next -> data << ' ';
      else
            cout << head -> data << ' ';
      recursionP (head->next);
      cout <<  head->data << ' ';
}
```

Output: _____

Q3. Consider the following pseudo-code? **(3 Marks)**

Output:

```
void f(int n, char s, char d, char t){
      if (n == 1){
            cout << s << ' '  << d << '\n';
            return;
      }
      f (n - 1 , s, t, d);
      cout << n << ' ' << s << ' '  << d << '\n';
      f (n - 1 , t, d, s);
}
f (3, 'A', 'C', 'B');
```

Working:

Q4 Suppose you are given a sorted array A of n distinct numbers that have been rotated k steps, where k is an unknow integer between 1 to n-1. The array is split such that the first k elements (A[0] to A[k-1]) and the remaining elements (A[k] to A[n-1]) are each sorted in increasing order, with the condition that A[n-1] < A[0]. Given this, describe or implement an approach to find the value of k in O(LogN). **(10 Marks)**

The array {9,13,16,18,19,23,28,31,37,42,0,1,2,5,7,8} has 16 elements with k =10.

Q5. Given two very long positive integers stored as ***Char*** type vectors . Write an ***add*** function to compute the sum of these integers and return the result as a new ***char*** type vector. Assume each integer is stored in either left-to-right(most significant to least significan digit) or right-to-left (least significant to most significant digit) order in the input vectors, and keep the same order in the resultant vector for consistency. **(10)**

Note 1: To secure full marks, write code-avoiding duplication.

```
vector<char> add(vector<char> &a, vector<char>& b){
```

Output:

```
  23182536475839542731
 985172263142849321282
----------------------
10083547996188688864013
```

Q6. In a doubly circular header-linked list, a ***swap*** method is used to **interchange** two disconnected nodes. The method does not include any **checks** or special handling for cases where the nodes to be **swapped are adjacent** (i.e., one node is the next or previous node of the other). Analyze if the current implementation of the swap method could cause any **issues** when swapping adjacent nodes, and explain why or why not. **(10)**

Note: Give brief to the point answer, which statement can cause the problem and why?

Q7. Write a brute-force method to evaluate an infix expression directly, without converting it to postfix or prefix notation. You may write either pseudocode, Python code, or C++ code. Also, discuss complexity of your code.**(10)**

```
input: "2+3*4"
output: 14
input: "(2+3)*4"
output: 20
```

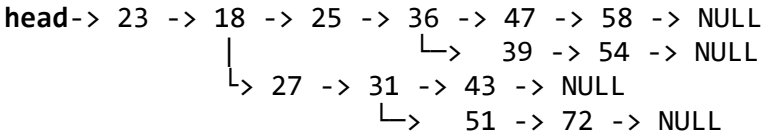
Q8 Write an efficient $O(NlogN)$ algorithm to generate a vector where each element at index i contains the count of elements in the original vector that are smaller than the element at index i. Specifically, for each element in the input vector, count the number of elements in the entire vector that are smaller, and store this count at the corresponding index in the output vector.

Note: An inefficient method will not be graded.

Example:

```
input: {3,7,1,2,4}
output: {2,0,4,3,1}
```

[**Bonus Question**] Consider class NNode with two pointers next1, next2. In NList nodes n1 and n2 may point to NULL or some next node on different paths like:



Traverse the list and give output in the format:

Output:

```
23 18
      25 36
            47  58
            39  54
      27 31
            43
            51 72
```