Data Structure Assignment

Q1. **Create a text file** data.txt for 100 student's records (as given in sample file segment) with appropriate headers: Name, ID, Quiz 1 (Out of 25), Quiz 2 (Out of 25), Quiz 3 (Out of 25), and Quiz 4 (Out of 25) as indicated into sample file below. Insert 0 if a student missed the quiz. Based on the file data.txt, create a program that reads the file and outputs its content to the screen, with the appropriate headers (ex: Name, ID, Quiz 1, Quiz 2, Quiz 3, Quiz 4). Allow the user to view the data in chunks, asking him to press any key in order to see the next chunk (of n records where n is supposed to be entered by user). At the end, print a statistics table, showing the average, lowest, and highest scores on each quiz/exam.

Sample File:

Name	ID	Quiz 1 (Out of 25)	Quiz 2 (Out of 25)	Quiz 3 (Out of 25)	Quiz 4 (Out of 25)
ABC	123	15	18	17	19

Perform following operations:

a. Display name and id of student who scored highest marks in Quiz 1.

b. Display name and id of students who scored more than 70% marks in all quizzes (i.e. Quiz 1 > 70%, Quiz 2 > 70% and so on).

c. Display name and id of students who scored overall (Quiz 1 + Quiz 2 + Quiz 3 + Quiz 4) more than 70% marks.

d. Identify whether there is a student or not who scored highest marks in each quiz (i.e. Let ABC is the student who scored highest marks in Quiz 1, Quiz 2, Quiz 3, and Quiz 4 then display his name.

Q2. Write a function that reverses the order of the nodes in a singly linked list. The function should have the prototype

Struct NODE *sll_reverse(struct NODE *first);

Declare the node structure in the file singly_linked_list_node.h.

The argument points to the first node in the list. After the list is rearranged, the function returns a pointer to a new head of the list. The link field of the last node in the list will contains NULL, and in the event of an empty list(first==NULL) the function should return NULL.

Q3. **Write a program** to identify second largest number in LL1 (**Do not sort the link list**).

Q4. Consider chain as Simple Linked List. Let x = (x1, x2, x3, ……., xn) and y = (y1, y2, y3, ……., ym) be two chains. Write a C function to merge the two chains together to obtain the Chain z = (x1, y1, x2, y2,……, xm , ym, xm+1,……. , xn) if m ≤ n and z = (x1, y1, x2, y2,……, xn, yn, xn+1,……. , xm) if m > n. Following the merge, x and y should represent empty chains because each node initially in x or y is now in z. No additional nodes may be used.

Q5. You have given a single link list of n numbers. You do not know n. Due to some reason the next pointer of a node instead of pointing to next element, it points to any node backward to that node and create a cycle. Due to this cycle when you traverse the link list it will not terminate. Identify the node which causes the cycle.

Q6. Let a and b be two polynomials represented as circular lists with header nodes. Write a C function to compute the product polynomial c = a*b. Your code should leave a and b unaltered. Write a C function to evaluate a polynomial at the point x, where x is a real number. Assume that the polynomial is represented as a circularly linked list with a header node.

Q7. Write a program to remove a node from a doubly linked list. The function should have the prototype:
int dll_remove(struct NODE *rootp, struct NODE *node);
You may assume that the node structure is defined in the file doubly linked_list_node.h. The first argument points to a node containing the root pointers for the list, and the second list to the node that is to be removed.
The function returns false if the list does not contain the indicated node, otherwise it removes the node and returns true.

Q8. Write a function to insert a new word into the concordance list. A concordance list is an alphabetic list of the words that appear in a book or article. The function should take a pointer to the list pointer and a string as its arguments. The string is assumed to contain a single word. If the word is not already exist in the list, it should be copied into dynamically allocates memory and then inserted. The function should return true if the string was inserted; and false if the string already existed in the list, did not begin with the letter, or if anything else went wrong. The function should maintain the primary list in order by the letter of the node and the secondary lists in order by the words.

Q9. Define a linked list with a loop as a linked list in which the tail element points to one of the list's elements and not to NULL. Assume that you are given a linked list L, and two pointers P1, P2 to the head. Write an algorithm that decides whether the list has a loop without modifying the original list.

Q10. Using only three linked list (source, sorted, and losers), sort a random sequence of numbers by first placing the numbers on the-source linked list. Assume the number on the top of the source is the largest, and push it on the sorted linked list. Continue to pop the source-linked list comparing it with the top of the sorted linked list. Whichever number is the smallest, pop it from its linked list and push it onto the on the-losers linked list. Once the source linked list is empty, repeat the process using the losers linked list as the source linked list, and use the source linked list as the losers linked list. The algorithm completes when all the numbers have been placed into the winners linked list.

Q11. You are given an array A of n sorted numbers that has been circularly shifted K positions to the right. For example, {35, 42, 5, 15, 27, 29} is a sorted array that has been circularly shifted K = 2 positions, while {27, 29, 35, 42, 5, 15} has been shifted K = 4 positions. Write a program to find the largest number in A if **(a)** K is known **(b)** K is unknown.

Q12. Write a program to read a list of students from a file and create a linked list. Each entry in the linked list should have the student's name, a pointer to the next pointer, and a pointer to a linked list of scores. There may be up to 4 scores for each student. The program should initialize the student list by reading the students' names from the file and creating null score lists.

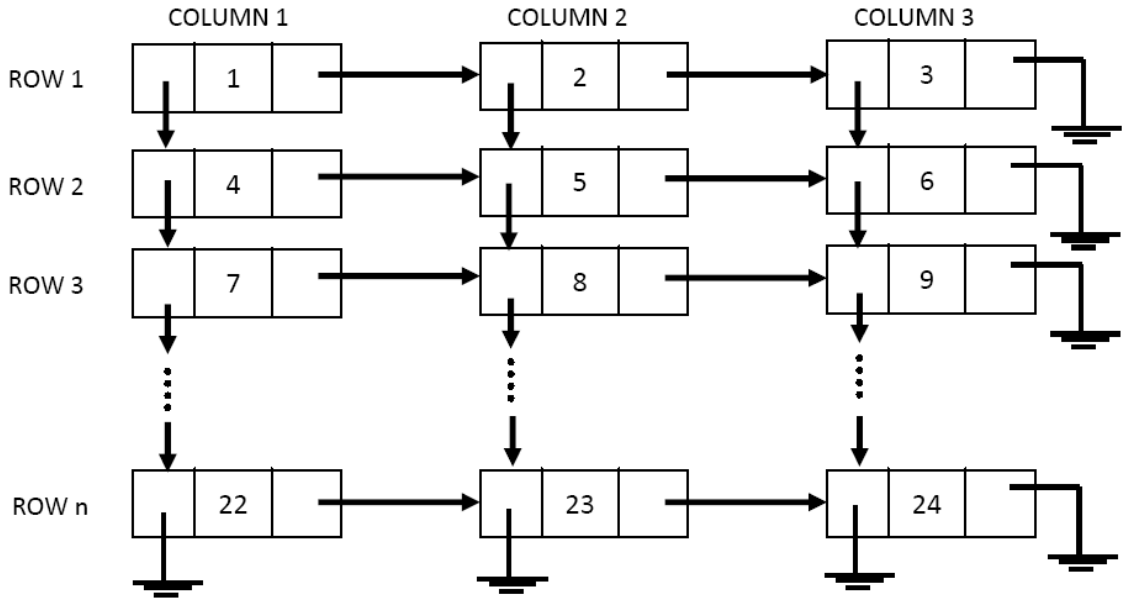
It should then loop through the list, prompting the user to enter the scores for each student. The scores' prompt should include the name of the student. After all scores have been entered, the program should print the scores for each student along with the score total and avg. The average should include only those scores present

Q13. Write a program to create a data structure as indicated in following figure. Further display the contents of this structure (a) Row wise (b) Column Wise.

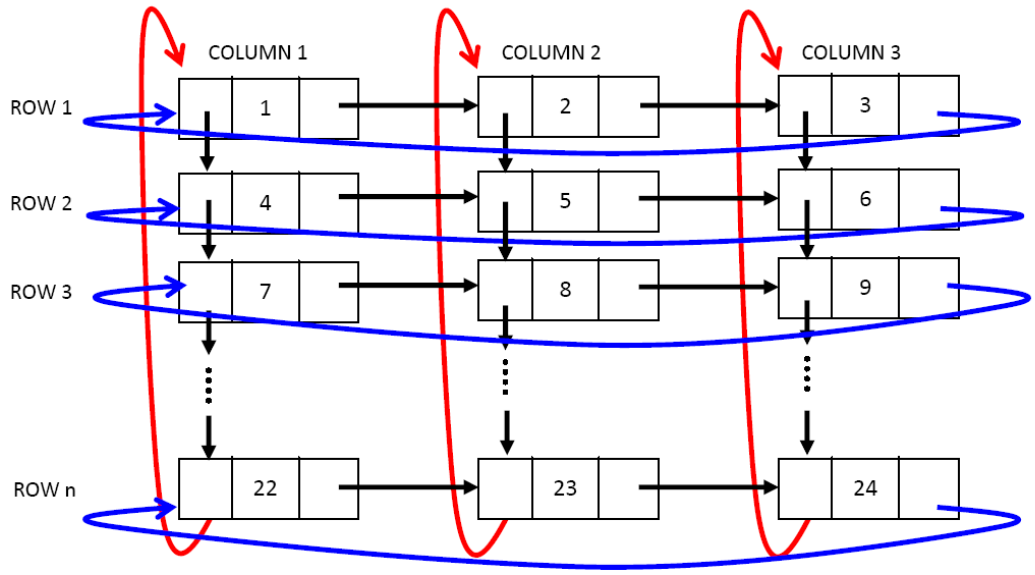
Struct nodeA

DP	**DATA**	**RP**

{
int Data;
struct nodeA *DP; struct nodeA *RP;};



Q14. Modify your program to create a data structure as indicated in following figure. Further display the contents of this structure (a) Row wise (b) Column Wise.



Q15. There are n books in a library (assuming that you do not know n). Each book has Book Id Number, Book Title, One main author (A book can be written by many author but only one author will be main author), and Name of Publisher. There can be multiple copies of single title book. Books are identified by Book Id Number (there will be unique book id even if there are multiple copies of single titled book). These Book Id's are 6 digit numbers starting from 100001 onwards. Each book has issue slip of m fixed records.
These records have following fields: Issue Date, Return Date, and Enrolment number of student to whom the book is currently issued. If one issue slip is completely filled then another issue slip of m records is attached with that book along with previous one.
Propose an efficient data structure if (a) n is known (b) if n is unknown, such that following queries can be answered efficiently.

(i)Display the list of books which are currently issued.

(ii)If one requires a book then he/she is supposed to enter book title. Display that the entered book is available in the library or not (remember that a single titled book can have multiple copies but different book id number).

(iii)Assuming that a book will not be issued to a student if there are 6 books already issued to him/ her. Now if there is a request from a student to get a book issued, then display the message whether he/she is entitled for that or not.

(iv)Assuming that if book is not returned by a student by its due date there will be late fine of Rs. 5 per day. Display the name of student who is supposed to pay maximum late fine.

(v)Due to rough handling of books, it is required to remove the book from Library. Therefore delete the records of that book from the list of books.

(vi)Library authorities keep on purchasing books at regular intervals and assign Book Id to each newly purchased book. Allotment of Book Id's are based upon the fact that starting from Book Id number 100001 upto last book id no number is missing. Say due to rough handling

Book Id number 100025 records were deleted from the list of books. Then one of the arrived books will have Book Id as 100025.

Consider the following information in the figure about a Library Scenario.

```
Library has the following books:

Fielding Henry
    * Pasquin - checked out to Chapman Carl
    * The History of Tom Jones
Fitzgerald Edward
    * Selected Works
    * Euphranor - checked out to Brown Jim
Murdoch Iris
    * The Red and the Green - checked out to Brown Jim
    * Sartre
    * The Bell

The following people are using the library:

Brown Jim has the following books
    * Fitzgerald Edward, Euphranor
    * Murdoch Iris, The Red and the Green
Chapman Carl has the following books
    * Fielding Henry, Pasquin
Kowalski Stanislaus has no books
```

Information about the catalogue(Books and Authors) and Users is to be maintained in the form of a Multi Linked Data Structure as shown below in the figure. Write a program to maintain the Library information as shown and provide all necessary operations of addition, updation and deletion (of books , authors and users) as well as issue and return operation of a library.

Q16. Implement array based and Linked List based Stack with all operations when (a)size of stack is know (b) stack size is not known.

Q17. Write a program that determines whether an input string is a palindrome, i.e. whether it can be read the same way forward and backward. At each point, you can read only one character of the input string. Do not use an array to first store this string ,Consider using multiple stacks.

Q18. Write a program to convert a number from decimal notation to a number expressed in a number system whose base (or radix) is a number between 2 and 9. The conversion is performed by repetitious division by the base to which a number is being converted and then taking the remainders of division in the reverse order. For example, in converting to binary,

number 6 requires three such divisions: 6/2 = 3 remainder 0, 3/2 = 1 remainder 1, and finally, 1/2 = 0 remainder 1. The remainders 0, 1, and 1 are put in the reverse order so that binary equivalent of 6 is equal to 110.

Q19. Write a program to convert (a) given postfix to prefix (b) given prefix to postfix (c) given infix to
postfix and further evaluate it to obtain the computed value.
(4 + 9 * 6) - ((8 – 6) / 2 * 4) * 9 / 3

Q20. Debug

```
void rearrange(struct node *odd)
{ if (odd == NULL || odd->next == NULL || odd->next->next == NULL)
return;
struct node *even = odd->next;
//Add line
odd = odd->next;
even->next = NULL;
while (odd && odd->next)
{
struct node *temp = odd->next->next;
//Add 2 lines
odd->next = temp;
if (temp != NULL)
odd = temp;
}
odd->next = even;
}
```

Q2.

```
void pairWiseSwap(struct node **head)
{
if (*head == NULL || (*head)->next == NULL)
return;
struct node *prev = *head;
struct node *curr = (*head)->next;
*head = curr;
while (true)
{
//Add line
curr->next = prev; // Change next of current as previous node
if (next == NULL || next->next == NULL)
{
prev->next = next;
break;
}
prev->next = next->next;
//Add line
curr = prev->next;
}
}
```

Q3.

```
void merge(struct node *p, struct node **q)
{
struct node *p_curr = p, *q_curr = *q;
struct node *p_next, *q_next;
while (p_curr != NULL && q_curr != NULL)
{
p_next = p_curr->next;
//Add line
q_curr->next = p_next; // Change next pointer of q_curr
p_curr->next = q_curr; // Change next pointer of p_curr
//Add line
q_curr = q_next;
}
*q = q_curr; // Update head pointer of second list
}
```

Q4.

```
void skipMdeleteN(struct node *head, int M, int N)
{
struct node *curr = head, *t;
int count;
while (curr)
{
for (count = 1; count<M && curr!= NULL; count++)
curr = curr->next;
if (curr == NULL)
return;
//Add line
for (count = 1; count<=N && t!= NULL; count++)
{
struct node *temp = t;
t = t->next;
free(temp);
}
//Add line
curr = t;
}
}
```

Q5.

```
void swapKth(struct node **head_ref, int k)
{
int n = countNodes(*head_ref);
if (n < k) return;
if (2*k - 1 == n) return;
//Add line
node *x_prev = NULL;
for (int i = 1; i < k; i++)
{
x_prev = x;
x = x->next;
```

```
}
node *y = *head_ref;
node *y_prev = NULL;
for (int i = 1; i < n-k+1; i++)
{
y_prev = y;
y = y->next;
}
if (x_prev)
//Add line
if (y_prev)
y_prev->next = x;
node *temp = x->next;
x->next = y->next;
//Add line
if (k == 1)
*head_ref = y;
if (k == n)
*head_ref = x;}
```