**Assignment-01(Marks 40)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | You have been provided with an array containing N numbers within the range of 0–100 (exclusive). Write the code to determine the mode of these numbers. In statistics, mode is defined as the value that appears most frequently in a given set of data. If multiple numbers have the same occurrence, anyone can be chosen as the mode.  **Note: Your code must follow the time complexity O(n).**   |  |  | | --- | --- | | Sample Input | Sample Output | | 7  3 2 5 10 5 1 7 | 5 | | 7  3 2 5 10 5 3 7 | 3 or 5 | |
| 2. | Your task is to implement functions for the following tasks in a singly linked list. You can use the code of the single linked list provided as a template in your class. **1. Insert into mid position:** This function will take an integer value and insert it in the middle of a linked list. You will have to count the number of elements in a LL to calculate the mid position. **2. Add all elements of an array at the end of the LL:** You will be given an array of integers. Your task is to add all the elements of the array at the end of the linked list. **3. Delete k-th element of a LL:** The user will provide the value of k. Your task is to delete the k-th element from the LL. You can traverse the whole LL only once. **4. Replace the last element of a LL using the sum of the LL:** Your task is to calculate the sum of all the elements in the LL and replace the last element of the LL by the value of the sum. **5. Search all the even numbers of a LL:** Your task is to traverse the LL and look for the even numbers and put them in an array.  **6. Reverse the LL**: Your task is to reverse the linked list. |
| 3. | Implement the following for a Singly Linked List that stores integers.   1. Create necessary structures 2. It will have both a **head** and a **middle**. 3. It will have 5 functions: printEvenList(), printList(), listLength(), insertAtMiddle(), and deleteMax(). 4. **printEvenList()** will print out only the even node with **O(n)**. 5. **printList()** will print out the entire list with **O(n)**. 6. **listLength()** will return the length of the list with **O(1)**. 7. **insertAtMiddle()** will take the item to be inserted and insert it at the middle of the list using middle pointer. For example, suppose the list is **1🡪6**. After calling **insertAtMiddle(2),** the list will be **1🡪2🡪6**. Again if we call **insertAtMiddle(3),** the list will be 1🡪2🡪3🡪6. **Don’t use loop**. 8. **deleteMax()** will delete the maximum element from the list. For example, suppose the list is **45🡪12🡪3🡪69🡪7**. After calling deleteMax(), the list will be **45🡪12🡪3🡪7**.   **For operations like insert and delete, remember that you may need to update the head and/or middle pointer.**  Implement a **main** function equivalent to the following main function to test your functions.   |  | | --- | | int main(){  insertAtMiddle(5);  insertAtMiddle(2);  insertAtMiddle(4);  insertAtMiddle(3);  insertAtMiddle(7);  printList(); // 5🡪4🡪7🡪3🡪2  printEvenList(); // 4🡪2  listLength(); //5  cout<<"Max is: "<<deleteMax()<<endl; // Max is: 7  printList(); // 5🡪4🡪3🡪2  return 0;  } | |
| 4. | The Stack will be implemented using **Linked List**. Your code will have the following functions: i) int pop() : Pops and returns the top element of the stack, the element is removed from the stack ii) void push(int elem) : Insert the given element inside the stack in the nth toppest position. Where n = (summation of all the digits of your Student ID % 4) + 1 iii) int isEmpty() : Return 1 if the Stack is empty, return 0 if the Stack is not empty iv) int peek() : Return the top element without removing it from the Stack v) void display() : Print all the elements of the Stack vi) void sort() :Add a function to sort the stack in ascending order (with biggest items on top) using only push, pop, peek operations. |
| 5. | Implement a priority queue using linked list. Your code will have the following functions:   1. Put(int val): It adds the value 'val' in such a manner that the list will be sorted in ascending order after each 'put' operation. 2. int pop(): it returns and delete the top item in the list. |
| 6. | Write a program to show the vertex with the highest number of adjacent nodes in an undirected graph. The first two lines of the input will contain v (vertices) and e (edges). The next e number of lines contains the list of edges. You have to create the Adjacency Matrix first then do the rest of the code. |
| 7. | Find the cost of the longest path from the source vertex to all other vertices in the graph, given a weighted directed acyclic graph (DAG) and a source vertex. If the vertices are inaccessible from the supplied source vertex, their distance is displayed as infinity. For example, consider the following DAG,    The longest distance of source vertex 7 to every other vertex is: dist(7, 0) = 7 (7 —> 3 —> 0) dist(7, 1) = -1 (7 —> 1) dist(7, 2) = -5 (7 —> 1 —> 2) dist(7, 3) = 4 (7 —> 3) dist(7, 4) = 9 (7 —> 3 —> 4) dist(7, 5) = -4 (7 —> 5) dist(7, 6) = 9 (7 —> 3 —> 0 —> 6) |
| 8. | Carbon Ltd. company is storing their employee details using a linked list.The employees have [name, id, position]. Here each consecutive employee has connection to each other.  Return the number of connected employees in number, where two entity are connected if they appear consecutively in the linked list.   |  |  | | --- | --- | | Input | Output | | Rana, 11023, Software-Tester  Salman, 12045, Designer  Jacob, 21055, FrontEnd-Developer  Rasel, 22134, BackEnd-Developer  [Rana, Jacob, Rasel] | 1 (Jacob, Rasel) | | Rana, 11023, Software-Tester  Salman, 12045, Designer  Jacob, 21055, FrontEnd-Developer  Rasel, 22134, BackEnd-Developer  Patrick, 31022, Administrator  [Rana, Rasel, Salman, Patrick] | 2 (Rana, Salman and Rasel, Patrick) | |
| 9. | Write a program which will take an integer N (0<N<1000) from the user. Generate N random numbers (range: 1-100) and store them into queue. Now write a function which will print the value that occurred most of the time. If multiple values occurred maximum time, then print all the values with number of occurrence.   |  |  | | --- | --- | | input | output | | N = 5  Random values: 12 8 25 9 25 | 25 occurred 2 times | | N = 7  Random values: 25 8 25 9 25 9 9 | 9, 25 occurred 3 times | |
| 10. | Carbon Ltd. company now wants to design a stack that supports push, pop, top, and retrieving the employee with minimum work-hour in constant time.  Implement the **MinStack** class:  declare an employee obj named **Emp** where each employee has **[name, id, work-hour]**  **MinStack()** initializes the stack object.  **void push(Emp employee)** pushes the object employee onto the stack.  **void pop()** removes the employee on the top of the stack.  **int top()** gets the top employee of the stack.  **int getMin()** retrieves the employee detail who has the least work-hour time in the stack.   |  |  | | --- | --- | | input | output | | N = 3  Rasel, 014, 45  Yaseer, 023, 55  Naheed, 033, 12 |  | |

Instructions:

1. You will get zero, if there are any plagiarism.
2. Submit ten .cpp files for ten problems in a single zip file.