**Bahria University, Lahore Campus**

Department of Computer Science

**Assignment 02**

**(Spring 2023)**

|  |  |  |
| --- | --- | --- |
| Course: | **Data Structure & Algorithms** | Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Course Code: | CSC-221 | Max Marks: **10** |
| Faculty’s Name: | Ms. Zupash Awais | **Deadline: 6th April 2023(11: 00 PM)** |

Name: \_AFFAN AHMAD\_\_\_ Enroll No: \_03-134221-003\_\_ Section: BS(cs)3A\_\_\_\_

Instructions:

1. Understanding each question is a part of the assignment.
2. Solve the assignment with full honesty. Leave the question that is not clear to you or you are not able to solve, we will discuss it in the class for better understanding.
3. Try to think of a solution of your own. If you will keep on searching for solutions on Google, it will not make your problem-solving ability strong. This assignment is to sharpen your mind a bit or make you able to solve/think of a solution for any problem.
4. Copied assignments (from each other) will be awarded ZERO.
5. Trying ChatGPT is totally prohibited. It won’t be available for you in Exam so don’t put your future in DANGER.
6. *The assignment is to be submitted in the soft form on LMS till the deadline mentioned above. Bring hardcopy (printed) of your solved assignment on Friday(7th April 2023)*
7. No need to take a colored print or bring the assignment in files. Submit it as a normal black and white stapled paper

***\*\*Provide the solution in the box for each question\*\****

**Question 1 Marks: 10**

Use the given singly linked list code given in the class and add the following functions in the list class.

* **ReverseDisplay**
* **CountAllNodes**
* **InsertionSort (Sort the List)**
* **SearchAny (Use Binary Search)**
* **UpdateAny**
* **DeleteAlternateNodes (delete alternate nodes of the list)**
* **CheckPositiveNodes**
* **CheckNegativeNodes**
* **SplitTheList (Split the list to make two Sperate lists, one of positive and one of negative nodes the nodes should be placed in a sorted manner in both lists)**
* **MergeLists (Merge the splitted lists and make one single sorted list)**

|  |
| --- |
| **Code** |
| **#include<iostream>**  **#include <stdlib.h>**  **using namespace std;**  **int t = 0;**  **int i=0,j=1;**  **struct node**  **{**  **int data;**  **node \*next;**  **node \*head, \*tail;**  **node \*head1,\*tail1;**  **node \*head2,\*tail2;**    **};**  **node\*mid;**  **node \*head = NULL, \*tail = NULL;**  **node \*head1 = NULL, \*tail1 = NULL;**  **node \*head2 = NULL, \*tail2 = NULL;**  **node \*even =NULL,\*odd=NULL;**  **void**  **insert\_start (int value)**  **node \*temp = new node;**  **temp->data = value;**  **temp->next = NULL;**  **if (head == NULL)**  **{**  **head = temp;**  **tail = temp;**  **}**  **else**  **{**  **temp->next = head;**  **head = temp;**  **}**  **}**  **void**  **display ()**  **{**  **node \*temp;**  **temp = head;**  **while (temp != NULL)**  **{**  **cout << temp->data << " ";**  **temp = temp->next;**  **}**  **}**  **void reverse()**  **{**    **node\* cur = head;**  **node \*pre = NULL, \*next = NULL;**    **while (cur != NULL) {**    **next = cur->next;**  **cur->next = pre;**    **pre = cur;**  **cur = next;**  **}**  **head = pre;**  **}**    **void sorting()**  **{**  **node \*cur;**  **node \*pre;**  **int temp ;**  **cur=head;**  **//temp=head;**  **while(cur!=NULL)**  **{**    **pre=cur->next;**  **while(pre!=NULL)**  **{**    **if (cur->data>pre->data)**    **{**  **temp=cur->data;**  **cur->data=pre->data;**  **pre->data=temp;**  **}**  **pre=pre->next;**    **}**  **cur=cur->next;**  **}**      **}**  **void check()**  **{**  **node \*temp;**  **temp=head;**  **while (temp!=NULL)**    **{**  **if (temp->data>0)**  **{**  **cout <<"your "<< j<<" node is positive "<< endl;**  **j++;**  **}**  **else**  **{**  **cout <<"your "<< j<<" node is negative "<< endl;**  **j++;**  **}**  **temp=temp->next;**  **}**    **}**  **struct node\* middle(struct node\* start,struct node\* last)**  **{**  **if (start == NULL)**  **return NULL;**    **struct node\* slow = start;**  **struct node\* fast = start -> next;**    **while (fast != last)**  **{**  **fast = fast -> next;**  **if (fast != last)**  **{**  **slow = slow -> next;**  **fast = fast -> next;**  **}**  **}**    **return slow;**  **}**    **void binarySearch( int value)**  **{**  **int temp;**  **struct node\* start = head;**  **struct node\* last = NULL;**    **do**  **{**    **struct node\* mid = middle(start, last);**      **if (mid == NULL)**  **{cout <<"list empty ";}**    **// If value is present at middle**  **if (mid -> data == value)**  **{ cout <<mid->data;**  **}**      **else if (mid -> data < value)**  **{ start = mid -> next;}**      **else**  **{last = mid;}**    **} while (last == NULL ||**  **last != start);**  **}**  **void update\_data(int old, int n) {**  **int pos = 0;**    **if(head==NULL) {**    **cout <<"list is empty "<< endl;**  **}**  **node\* current = head;**  **while(current->next!=NULL) {**  **if(current->data == old) {**  **current->data = n;**  **cout <<old<<"found at position "<<pos<<" replaced with "<< n << endl;**  **break;**    **}**    **current = current->next;**  **pos++;**  **}**  **}**  **void split\_merge()**  **{**  **node \*temp;**  **node \*temp1 = new node;**  **node \*temp2 = new node;**  **temp=head;**  **while (temp!=NULL)**    **{**  **if (temp->data>0)**  **{**    **temp1->data = temp->data;**  **temp1->next = NULL;**  **if (head1 == NULL)**  **{**  **head1 = temp1;**  **tail1 = temp1;**  **}**  **else**  **{**  **temp1->next = head;**  **head = temp1;**  **}**  **}**  **if(temp->data<0)**  **{**    **temp2->data = temp->data;**  **temp2->next = NULL;**  **if (head2 == NULL)**  **{**  **head2 = temp2;**  **tail2 = temp2;**  **}**  **else**  **{**  **temp2->next = head2;**  **head2 = temp2;**  **}**  **}**  **temp=temp->next;**  **}**  **cout <<"your positive nodes is :";**  **node \*temp3;**  **temp3=head1;**  **while(temp3!=NULL)**  **{**  **cout <<temp3->data<<" ";**  **temp3=temp3->next;**  **}**  **cout <<"your negative nodes is :";**  **node \*temp4;**  **temp4=head2;**  **tail2->next=NULL;**  **while(temp4!=NULL)**  **{**  **cout <<temp4->data<<" ";**  **temp4=temp4->next;**  **// break;**  **}**  **cout <<endl;**  **//merge**  **cout <<"merge list is :"<< endl;**  **node \*temp5;**  **node \*cur;**  **cur=head1;**  **temp5=head1;**  **while(cur->next!=NULL)**  **{**  **cur=cur->next;**  **}**  **cur->next=head2;**  **while(temp5!=NULL)**  **{**  **cout <<temp5->data<<" ";**  **temp5=temp5->next;**  **}**  **}**  **int main ()**  **{**  **int val, ch, pos;**  **cout << "press 1 for insert at start " << endl;**  **cout << "press 2 for display " << endl;**  **cout << "press 3 for revere display " << endl;**  **cout << "press 4 for total nodes " << endl;**  **cout << "press 5 for check nodes " << endl;**  **cout << "press 6 for sorting nodes " << endl;**  **cout << "press 7 for Search :" << endl;**  **cout << "press 8 for update\_data :" << endl;**  **cout << "press 9 for split\_merge :" << endl;**  **cout << "press 10 for Exit :" << endl;**  **cout <<endl;**  **do**  **{**  **cout << "Enter your choise :";**  **cin >> ch;**  **if (ch == 1)**  **{**  **++t;**  **cout << "Enter the value :";**  **cin >> val;**  **//createnode(val);**  **insert\_start (val);**  **}**  **if (ch == 2)**  **{**  **cout << "your list is " << endl;**  **display ();**  **cout << endl;**  **}**  **if (ch == 6)**  **{**  **cout << "now press 7 " << endl;**    **cout << endl;**  **}**  **if (ch == 4)**  **{**  **if (t <= 0)**  **{**  **cout << "list is empty " << endl;**  **}**  **else**  **{**  **cout << "the total nodes in linklist is :" << t << endl;**  **}**  **}**    **if (ch == 5)**  **{**    **check();**  **}**    **if (ch == 7)**  **{**  **cout << "your list is sorted now"<<endl ;**  **sorting();**  **}**  **if (ch == 8)**  **{**  **int p;**  **cout << "enter your value :";cin>>p;**  **binarySearch(p);**  **}**    **if (ch == 8)**  **{**  **int old ,n ;**  **cout <<"enter old data ";cin>> old;**  **cout <<"enter new data ";cin>> n;**    **update\_data(old,n);**    **}**  **if (ch == 9)**  **{**  **split\_merge();**  **}**  **if (ch == 10)**  **{**  **cout << "exit" << endl;**  **break;**  **}**  **}**  **while (ch != 10);**  **return 0;**  **}** |
| **Output:** |
|  |