|  |
| --- |
| **Data Structures and Algorithms**  ***Section*: BSCE2021 Assignment # 7 *Total marks*: 100**  ***Name*** : ***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll number* : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

***Submission:***

• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.*

*• Submission after due time will not be accepted.*

**There should be a Report explaining your code and highlighting the results. Follow this naming convention for your report RollNumber\_Assignment#.pdf e.g BSCE21000\_Assignment7.pdf.**

**TASK:**

Implement the following Sorts using Linked list

Selection Sort

Bubble Sort

Program should be menu driven. the program should continue to run until the exit option is selected. Handle all corner cases.

|  |
| --- |
| **FUNCTION.H:**  #include <iostream> using namespace std;  class Node { //made a class of node public:  int data; //declared data in public  Node \*nextPtr;   Node(int val) {  data = val; //made a constructor to set values  nextPtr = NULL;  }   void setNextPtr(Node \*n) {  nextPtr = n; //set the next ptr  }   Node \*getNextPtr() {  return nextPtr; //get the value of next ptr  } };  class linkList { //making 1 other class of linklist public:  Node \*tail;  Node \*head; //made some pointers  linkList() {  head = nullptr; //declared them to zero  tail = nullptr;  }  void takingInput(int value){  cout<<"IF YOU WANT TO STOP ADDING VALUES THEN ENTER -1."<<endl;  do{  cout<<"ENTER VALUE = ";  cin>>value;  if(value==-1){  cout<<"YOU CHOOSE TO BREAK."<<endl;  break;  }  else{  Node \*temp = new Node(value); //declaring  temp->data = value;  temp->nextPtr = nullptr; //initializing the next ptr in the next of the new initialized node  if (head == nullptr) {  head = temp; //if the head is null then store the temp in head  } else {  Node \*temp1 = head; //else make a new node  while (temp1->nextPtr != nullptr) //iterate it till the node is not null  temp1 = temp1->nextPtr; //store the temp to next ptr address  temp1->nextPtr = temp; //store pointer to the last one  }  }  }while(value!=-1);   }   void bubbleSort(){  Node \*i; //declaring the nodes  Node \*j;  int temp; //declaring temp for the swapping  for(i=head;i!= nullptr;i=i->nextPtr){ //a loop for iterations until the i is not null and the i is head  for(j=i->nextPtr;j!= nullptr;j=j->nextPtr){ //loop starting from previous index of i and moving till it is null  if(i->data>j->data){ //comparing the values of i and j  temp=i->data; // if i is greater(>), then swapping otherwise move the loop  i->data=j->data;  j->data=temp;  }  }  }  }  void selectionSort(){  Node \*temp=head; //initializing node with head  int swap; //declaring variable for swapping  Node \*min; //declaring nodes  Node \*temp1;  while(temp!= nullptr){ //checking if the temp is not null  min=temp; //then coming in and initializing min with the temp that is pointing towards head  temp1=temp->nextPtr; //and temp1 with the next index of temp  while(temp1!= nullptr){ //checking if the next index of temp that is temp1 is not null  if(min->data>temp1->data){ //comparing the temp node and the node after it  min=temp1; //if min->temp which is greater than reinitialize it with the temp1  }  temp1=temp1->nextPtr; //temp1 pointing to next index  }  swap=min->data; //swapping min and temp  min->data=temp->data;  temp->data=swap;  temp=temp->nextPtr; //pointing to the next index  }  }  void display() {  Node \*temp = head; //declaring  while (temp != nullptr) {  cout << temp->data << "\t"; //displaying the data  temp = temp->nextPtr; //storing the next address  cout << " "; //displaying space  }  }   };  **MAIN.CPP:**  #include <iostream> #include "Functions.h" using namespace std; // int main() {   Node n(5); //making object  n.setNextPtr(0);  n.getNextPtr();  linkList l; //making object  int opt;  cout<<endl;  cout << "\nENTER OPTION." << endl;  cout << "1.BUBBLE SORT." << endl;  cout << "2.SELECTION SORT." << endl;  cout << "3.EXIT." << endl; //taking input  cin >> opt;  if (opt == 1) {  int value;  l.takingInput(value); //storing in the link list  cout << "BEFORE SWAPPING = ";  l.display(); //displaying before swapping  cout << endl;  l.bubbleSort(); //calling bubble sort function  cout << "AFTER SWAPPING = ";  l.display(); //displaying after swapping  }  if (opt == 2) {  int value;  l.takingInput(value); //storing in the link list  cout << "BEFORE SWAPPING = ";  l.display(); //displaying before swapping  cout << endl;  l.selectionSort(); //calling the selection sort function  cout << "AFTER SWAPPING = ";  l.display(); //displaying after swapping  }  if (opt == 3) {  cout << "YOU CHOOSE TO EXIT." << endl;  exit(3);  }  return 0;  }  **OUTPUT:Text  Description automatically generated** |