### 

# **NATIONAL INSTITUTE OF TECHNOLOGY RAIPUR**

### 

## **B.Tech. (5th Semester)**

## **Assignment No :- 3**

## **Department of Computer Science & Engineering**

## 

## **Subject: Advance data Structure**

## **Lab Code- CS105201CS**

## 

## **Date:- 11/08/2025**

### **Submitted By :- Jayesh Sharma**

### **Roll No :- 23115045**

### **Lab Batch No :- 1**

### 

### **Q.1) Write a program that builds two linked lists and then append the second one to the end of the first one :-**

| #include <iostream> using namespace std;  struct Node {  int data;  Node\* next;  Node(int val) : data(val), next(nullptr) {} };  void append(Node\*& head, int val) {  Node\* newNode = new Node(val);  if (!head) { head = newNode; return; }  Node\* temp = head;  while (temp->next) temp = temp->next;  temp->next = newNode; }  void appendList(Node\*& head1, Node\* head2) {  if (!head1) { head1 = head2; return; }  Node\* temp = head1;  while (temp->next) temp = temp->next;  temp->next = head2; }  void printList(Node\* head) {  while (head) {  cout << head->data << " -> ";  head = head->next;  }  cout << "NULL\n"; }  int main() {  Node\* list1 = nullptr;  Node\* list2 = nullptr;   append(list1, 1);  append(list1, 2);  append(list1, 3);   append(list2, 4);  append(list2, 5);  append(list2, 6);   cout << "List 1: "; printList(list1);  cout << "List 2: "; printList(list2);   appendList(list1, list2);   cout << "After appending List 2 to List 1: ";  printList(list1); } |
| --- |

### **Q.2) Develop a program to build a singly linked list of 4 nodes and each node consists of character data value :-**

| #include <iostream> using namespace std;  struct Node {  char data;  Node\* next;  Node(char val) : data(val), next(nullptr) {} };  void append(Node\*& head, char val) {  Node\* newNode = new Node(val);  if (!head) { head = newNode; return; }  Node\* temp = head;  while (temp->next) temp = temp->next;  temp->next = newNode; }  void printList(Node\* head) {  while (head) {  cout << head->data << " -> ";  head = head->next;  }  cout << "NULL\n"; }  int main() {  Node\* head = nullptr;   append(head, 'A');  append(head, 'B');  append(head, 'C');  append(head, 'D');   cout << "Singly linked list: ";  printList(head); } |
| --- |

### **Q.3) Write a program to print all the data values in the above linked list :-**

| #include <iostream> using namespace std;  struct Node {  char data;  Node\* next;  Node(char val) : data(val), next(nullptr) {} };  void append(Node\*& head, char val) {  Node\* newNode = new Node(val);  if (!head) { head = newNode; return; }  Node\* temp = head;  while (temp->next) temp = temp->next;  temp->next = newNode; }  void printList(Node\* head) {  while (head) {  cout << head->data << " ";  head = head->next;  }  cout << endl; }  int main() {  Node\* head = nullptr;   append(head, 'X');  append(head, 'Y');  append(head, 'Z');  append(head, 'W');   cout << "Linked list values: ";  printList(head); } |
| --- |

### **Q.4) Write a program to reverse the above linked list :-**

| #include <iostream> using namespace std;  struct Node {  int data;  Node\* next;  Node(int val) : data(val), next(nullptr) {} };  void append(Node\*& head, int val) {  Node\* newNode = new Node(val);  if (!head) { head = newNode; return; }  Node\* temp = head;  while (temp->next) temp = temp->next;  temp->next = newNode; }  void reverseList(Node\*& head) {  Node\* prev = nullptr;  Node\* curr = head;  Node\* next = nullptr;  while (curr) {  next = curr->next;  curr->next = prev;  prev = curr;  curr = next;  }  head = prev; }  void printList(Node\* head) {  while (head) {  cout << head->data << " -> ";  head = head->next;  }  cout << "NULL\n"; }  int main() {  Node\* head = nullptr;   append(head, 10);  append(head, 20);  append(head, 30);  append(head, 40);   cout << "Original list: ";  printList(head);   reverseList(head);   cout << "Reversed list: ";  printList(head); } |
| --- |

### **Q.5) Write a program to reverse the above linked list :-**

| #include <iostream> #include <set> using namespace std;  struct Node {  int data;  Node\* next;  Node(int val) : data(val), next(nullptr) {} };  void append(Node\*& head, int val) {  Node\* newNode = new Node(val);  if (!head) { head = newNode; return; }  Node\* temp = head;  while (temp->next) temp = temp->next;  temp->next = newNode; }  Node\* unionList(Node\* head1, Node\* head2) {  set<int> seen;  Node\* result = nullptr;  while (head1) {  if (seen.insert(head1->data).second)  append(result, head1->data);  head1 = head1->next;  }  while (head2) {  if (seen.insert(head2->data).second)  append(result, head2->data);  head2 = head2->next;  }  return result; }  void printList(Node\* head) {  while (head) {  cout << head->data << " -> ";  head = head->next;  }  cout << "NULL\n"; }  int main() {  Node\* list1 = nullptr;  Node\* list2 = nullptr;   append(list1, 1);  append(list1, 2);  append(list1, 3);   append(list2, 3);  append(list2, 4);  append(list2, 5);   cout << "List 1: "; printList(list1);  cout << "List 2: "; printList(list2);   Node\* unioned = unionList(list1, list2);   cout << "Union: ";  printList(unioned); } |
| --- |