|  |  |
| --- | --- |
| **Data Structures & Algorithms**  Diploma in CSF, IT  Year 2/3 (2020/21) Semester 4/6 | **Week 3** |
| **1-2 Hours** |
| **Tutorial 3 – Linked List** | |

1. Explain the purpose of the following operators

1. **&**

|  |
| --- |
| Pointer.  Get the memory address |

1. **\***

|  |
| --- |
| Dereference operator.  To be used with pointers. It gets the value referenced. |

1. **new**

|  |
| --- |
| It allocates memory. Memory is obtained from the heap space |

2. Assuming the Node structure is declared as:

struct Node

{

string item; // to store the data item

Node \*next; // pointer to point to next node

};

1. Draw diagrams to show what happens in computer memory when the following statements are executed.

Node node1;

Node \*node2;

|  |
| --- |
| Node node1;  Node1 -> Item Next |
| Node \*node2;  Node2 -> address of node1 |

1. Write the statements to:

(i) store “Kevin” in node1.

(ii) initialize the next pointer in node1 to NULL

|  |
| --- |
| Node1.item = “Kevin:  Node1.next = NULL |

1. Write the statements to:
2. create a Node object and set node2 to point to it
3. store “Vivian” in the Node object pointed to by node2.

(iii) initialize the next pointer in Node object, pointed to by node2, to NULL

|  |
| --- |
| Node node2 = new Node;  Node1.next = node2;  Node2.item = “Vivian”;  Node2.next = NULL; |

1. What happens when the following statements are executed?

Node node3 = node1;

|  |
| --- |
|  |

Node \*node4 = node2;

|  |
| --- |
|  |

3. The specification of a List ADT implemented using Pointers is given below.

|  |
| --- |
| *// List.h - - Specification of List ADT (implemented using Pointers)*  #pragma once  #include<string>  #include<iostream>  using namespace std;  typedef string ItemType;  class List  {  private:  struct Node  {  ItemType item; // item  Node \*next; // pointer pointing to next item  };  Node \*firstNode; // point to the first item  int size; // number of items in the list  public:  // constructor  List();  *// add an item to the back of the list (append)*  bool add(ItemType item);  *// add an item at a specified position in the list (insert)*  bool add(int index, ItemType item);  *// remove an item at a specified position in the list*  void remove(int index);  *// get an item at a specified position of the list (retrieve)*  ItemType get(int index);  *// check if the list is empty*  bool isEmpty();  *// check the size of the list*  int getLength();  }; |

Implement the following operations of the List ADT

(a) bool add(ItemType item)

|  |
| --- |
|  |

(b) bool add(int index, ItemType item);

|  |
| --- |
|  |

(c) int getLength()

|  |
| --- |
|  |