Name: **Enrolment No:**	UPES

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

End Semester Examination, December 2020

Programme Name: B.Tech (SoE)		**Semester : II**
Course Name	**: Data Structure (Minor Specialization)**	**Time : 3 Hr**
Course Code	**: MRDA0201**	**Max. Marks : 100**

Instructions:

1. **Attempt all the questions[MCQ]**
2. **Attempt all questions serially as per Question paper.**
3. **Answer should be neat and clean. Draw a free hand sketch for circuits/tables/schematics wherever required.**
4. **You are expected to be honest about each attempt which you make to progress in life**

S. No.		**Marks**	**CO**
Q1	A circular linked list can be used for: A – Stack B – Queue C – Both Stack & Queue D – Neither Stack or Queue	**2**	
Q2	A linear collection of data elements where the linear node is given by means of pointer is called? a) Linked list b) Node list c) Primitive list d) Unordered list	**2**	

Q3	In linked list each node contains a minimum of two fields. One field is data field to store the data second field is? a) Pointer to character b) Pointer to integer c) Pointer to node d) Node	**2**	
Q4	Assuming int is of 4bytes, what is the size of int arr[15];? a) 15 b) 19 c) 11 d) 60	**2**	
Q5	Circular Queue is also known as ________ a) Ring Buffer b) Square Buffer c) Rectangle Buffer d) Curve Buffer	**2**	
Q6	If the elements "A", "B", "C" and "D" are placed in a queue and are deleted one at a time, in what order will they be removed? a) ABCD b) DCBA c) DCAB d) ABDC	**2**	
Q7	A normal queue, if implemented using an array of size MAX_SIZE, gets full when? a) Rear = MAX_SIZE – 1 b) Front = (rear + 1)mod MAX_SIZE c) Front = rear + 1 d) Rear = front	**2**	

Q8	Queues serve major role in ______________? a) Simulation of recursion b) Simulation of arbitrary linked list c) Simulation of limited resource allocation d) Simulation of heap sort	**2**	
Q9	Which of the following is false about a doubly linked list? a) We can navigate in both the directions b) It requires more space than a singly linked list c) The insertion and deletion of a node take a bit longer d) Implementing a doubly linked list is easier than singly linked list	**2**	
Q10	What is a memory efficient double linked list? a) Each node has only one pointer to traverse the list back and forth b) The list has breakpoints for faster traversal c) An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list d) A doubly linked list that uses bitwise AND operator for storing addresses	**2**	

Q11	Which of the following piece of code removes the node from a given position?
	
	a)
	
	`public void remove(int pos)`
	`{`
	`        if(pos<0 || pos>=size)`
	`        {`
	`                System.out.println("Invalid position");`
	`                return;`
	`        }`
	`        else`
	`        {`
	`                if(head == null)`
	`                        return;`
		2	

```
                if(pos == 0)
                {
                        head = head.getNext();
                        if(head == null)
                        tail = null;
                }
                else
                {
                        Node temp = head;
                        for(int i=1; i<position; i++)
                        temp = temp.getNext();
                }
                temp.getNext().setPrev(temp.getPrev());
                temp.getPrev().setNext(temp.getNext());
        }
        size--;
}
```

b)

```
public void remove(int pos)
{
        if(pos<0 || pos>=size)
        {
                System.out.println("Invalid position");
                return;
        }
        else
        {
                if(head == null)
                return;
                if(pos == 0)
                {
                        head = head.getNext();
                        if(head == null)
                        tail = null;
                }
                else
                {
                        Node temp = head;
                        for(int i=1; i<position; i++)
                        temp = temp.getNext();
                }
                temp.getNext().setPrev(temp.getNext());
                temp.getPrev().setNext(temp.getPrev());
        }
        size--;
}
```

c)

```
public void remove(int pos)
{
        if(pos<0 || pos>=size)
        {
                System.out.println("Invalid position");
                return;
        }
        else
        {
                if(head == null)
                        return;
                if(pos == 0)
                {
                        head = head.getNext();
                        if(head == null)
                        tail = null;
                }
                else
                {
                        Node temp = head;
                        for(int i=1; i<position; i++)
                        temp = temp.getNext().getNext();
                }
                temp.getNext().setPrev(temp.getPrev());
                temp.getPrev().setNext(temp.getNext());
        }
        size--;
}
```

d)

```
public void remove(int pos)
{
        if(pos<0 || pos>=size)
        {
                System.out.println("Invalid position");
                return;
        }
        else
        {
                if(head == null)
                        return;
                if(pos == 0)
                {
                        head = head.getNext();
                        if(head == null)
                        tail = null;
                }
                else
                {
                        Node temp = head;
                        for(int i=1; i<position; i++)
                        temp = temp.getNext().getNext();
```

	```
	                }
	                temp.getNext().setPrev(temp.getNext());
	                temp.getPrev().setNext(temp.getPrev());
	        }
	        size--;
	}
	```
Q 12	Entries in a stack are "ordered". What is the meaning of this statement? a) A collection of stacks is sortable b) Stack entries may be compared with the '<' operation c) The entries are stored in a linked list d) There is a Sequential entry that is one by one	**2**	
Q13	Which of the following is not the application of stack? a) A parentheses balancing program b) Tracking of local variables at run time c) Compiler Syntax Analyzer d) Data Transfer between two asynchronous process	**2**	
Q14	Here is an infix expression: 4 + 3*(6*3-12). Suppose that we are using the usual stack algorithm to convert the expression from infix to postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression? a) 1 b) 2 c) 3 d) 4	**2**	

Q15	Let A be a square matrix of size n x n. Consider the following code. What is the expected output?
	```
	C = 100
	for i = 1 to n do
	    for j = 1 to n do
	    {
	        Temp = A[i][j] + C
	        A[i][j] = A[j][i]
	```
		2	

	`A[j][i] = Temp - C`
	`}`
	`for i = 1 to n do`
	`for j = 1 to n do`
	`Output(A[i][j]);`
	
	(a) Matrix A itself
	(b) Transpose of Matrix A
	(c) Adding 100 to the upper diagonal elements and subtracting 100 from diagonal elements of A
	(d) None of these		

Q16	Which data structure is needed to convert infix notation to postfix notation? (a) Branch (b) Tree (c) Queue (d) Stack	**2**	
Q17	A data structure in which elements can be inserted or deleted at/from both the ends but not in the middle is? (a) Queue (b) Circular queue (c) Dequeue (d) Priority queue	**2**	
Q18	Which of the following points is/are not true about Linked List data structure when it is compared with array? a) Arrays have better cache locality that can make them better in terms of performance b) It is easy to insert and delete elements in Linked List c) Random access is not allowed in a typical implementation of Linked Lists d) Access of elements in linked list takes less time than compared to arrays	**2**	
Q19	You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list? a) Delete the first element b) Insert a new element as a first element c) Delete the last element of the list d) Add a new element at the end of the list	**2**	
Q20	Which of the following application makes use of a circular linked list? a) Undo operation in a text editor b) Recursive function calls c) Allocating CPU to resources d) Implement Hash Tables	**2**	
Q21	Which of the following is not a stable sorting algorithm in its typical implementation? a) Selection Sort	**2**	

	b) Bubble Sort c) Merge Sort d) Insertion Sort		
Q22	Consider a situation where swap operation is very costly. Which of the following sorting algorithms should be preferred so that the numbers of swap operations are minimized in general? a) Merge Sort b) Selection Sort c) Insertion Sort d) Heap Sort	**2**	
Q23	What is the advantage of bubble sort over other sorting techniques? a) It is faster b) Consumes less memory c) Detects whether the input is already sorted d) All of the mentioned	**2**	
Q24	Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this: 2 5 1 7 9 12 11 10 Which statement is correct? a) The pivot could be either the 7 or the 9. b) The pivot could be the 7, but it is not the 9 c) The pivot is not the 7, but it could be the 9 d) Neither the 7 nor the 9 is the pivot.	**2**	
Q25	Which of the following don't use matrices? a) In solving linear equations b) Image processing c) Graph theory d) Sorting numbers	**2**	
Q26	Which of the following is the way to represent Sparse Matrix? (a) Array (b) Linked list (c) Both (a) and (b) (d) None of the above	**2**	

Q27	What is the output of following function for start pointing to first node of following linked list? 1->2->3->4->5->6
	
	`void fun(struct node* start)`
	`{`
	`   if(start == NULL)`
	`     return;`
	`   printf("%d ", start->data);`
	`   if(start->next != NULL )`
	`     fun(start->next->next);`
	`   printf("%d ", start->data);`
	`}`
		2	

	(a) 1 4 6 6 4 1 (b) 1 3 5 1 3 5 (c) 1 2 3 5 (d) 1 3 5 5 3 1		

Q28	What is the functionality of the following piece of code?
	
	`public int function(int data)`
	`{`
	`Node temp = head;`
	`int var = 0;`
	`while(temp != null)`
	`{`
	`if(temp.getData() == data)`
	`{`
	`return var;`
	`}`
	`var = var+1;`
	`temp = temp.getNext();`
	`}`
	`return Integer.MIN_VALUE;`
	`}`
	
	(a) Find and delete a given element in the list
	(b) Find and return the given element in the list
	(c) Find and return the position of the given element in the list
	(d) Find and insert a new element in the list
		2	

Q29	In circular linked list, insertion of node requires modification of? (a) One pointer (b) Two pointer (c) Three pointer (d) None	**2**	

Q30	Consider the following pseudocode that uses a stack. What is output for input "**letsfindc**"?
	
	`declare a stack of characters`
	`while ( there are more characters in the word to read )`
	`{`
	`read a character`
	`push the character on the stack`
	`}`
	`while ( the stack is not empty )`
	`{`
	`pop a character off the stack`
	`write the character to the screen`
	`}`
	
	(a) Letsfindcletsfindc
	(b) Cdnifstel
	(c) Letsfindc
	(d) cdnifstelcdnifstel
	(e) cdnifstel
		2	

Q31	Let the following circular queue can accommodate maximum six elements with the following data. What will happen after ADD 'O' operation takes place?
	
	`front = 2 rear = 4`
	
	`queue = _______; L, M, N, ___, ___`
	
	(a) front = 2 rear = 5
	queue = ______; L, M, N, O, ___
	
	(b) front = 3 rear = 5
	queue = L, M, N, O, ___
	
	(c) front = 3 rear = 4
	queue = ______; L, M, N, O, ___
	
	(d) front = 2 rear = 4
	queue = L, M, N, O, ___
		2	

Q32

Fill in the blanks named with **C1 to C5** owing to let the program display the following output and operate the stack appropriately:

1) Push in stack

2) Pop from stack"

3) Display stack

4) Exit

Enter Choice:__________________________

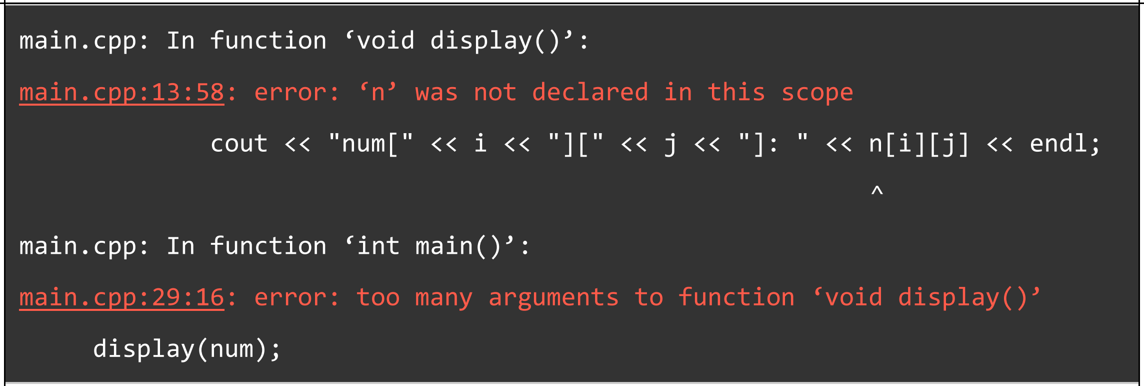
```
//Program starts from this line

#include <iostream>

using namespace std;


int stack[100], n=100, top = -1;


void push(int val)

{

  if(top>=n-1)

  cout<<"Stack Overflow"<<endl;

  else

  {

    //________________C1

    stack[top]=val;

  }

}
```

5

```
void pop()
{
  if(top<= -1)
  cout<<"Stack Underflow"<<endl;
  else {
    cout<<"The popped element is "<< stack[top] <<endl;
    // ___________________________C2
  }
}

void display()
{
  if(top>=0)
{
    cout<<"Stack elements are:";
    for(int i=top; i>=0; i--)
    cout<<stack[i]<<" ";
    cout<<endl;
  }
  else
   cout<< //___________________________C3
}

int main()
{
```

```
int ch, val;


cout<<"1) Push in stack"<<endl;

cout<<"2) Pop from stack"<<endl;

cout<<"3) Display stack"<<endl;

cout<<"4) Exit"<<endl;


do {

  cout<<"Enter choice: "<<endl;

  cin>>ch;

  switch( //__________________C4)

    {

    case 1:

    {

      cout<<"Enter value to be pushed:"<<endl;

      cin>>val;

      push(val);

      break;

    }

    case 2:

    {

      pop();

      break;

    }

    case 3:
      {
        display();
        break;
      }
      case 4:
      {
        cout<<"Exit"<<endl;
        break;
      }
      default:
      {
        cout<<"Invalid Choice"<<endl;
      }
    }
  }
  while(// ________________C5);
  return 0;
}
```

Q33



```
main.cpp: In function ‘void display()’:
main.cpp:13:58: error: ‘n’ was not declared in this scope
             cout << "num[" << i << "][" << j << "]: " << n[i][j] << endl;
                                                          ^
main.cpp: In function ‘int main()’:
main.cpp:29:16: error: too many arguments to function ‘void display()’
     display(num);
```

```
#include <iostream>
using namespace std;
```

3

```
void display() {

    cout << "Displaying Values: " << endl;

    for (int i = 0; i < 3; ++i) {

        for (int j = 0; j < 2; ++j) {

            cout << "num[" << i << "][" << j << "]: " << n[i][j] << endl;

        }

    }

}



int main() {

    int num[3][2] = {

        {3, 4},

        {9, 5},

        {7, 1}

    };

    display(num);



    return 0;

}
```

Above Error(represented in colored format) has occurred due to:

(a) Not passing a 1-D array as a function parameter
(b) Passing a 2-D Array as a function parameter
(c) Not declaring a 2-D Array
(d) Not passing a 2-D Array as a function parameter

Q 34	Correct the program at the indicated places while typing your answers followed by a comma:	**5**	

```
#include<iostream>

using namespace std;

_____________Student

{

   // Data Members

int roll = 34;

int age = 21;

int marks=79;

char name[20];


// Member Functions

void printDetails()

{

   cout<<"Roll = "<<roll<<"\n";

   cout<<"Age = "<<age<<"\n";

   cout<<"Marks = "<<marks;

}

}______________________


int main()

{

   struct Student S;

   cout<<" Enter your name\n";

   cin>>________________________
   cout<<"Enter Roll no. \n";

   cin>>________________________



  S.printDetails();



________________________________________



}
```

Q35

```
#include <bits/stdc++.h>
using namespace std;

// A linked list node
class Node
{
    public:
    int data;
    Node* next;
    Node* prev;
};


/* Given a reference (pointer to pointer) to the head of a list
and an int, inserts a new node on the front of the list. */

void push(Node** head_ref, int new_data)
{
    /* 1. allocate node */
    Node* new_node = new Node();

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. Make next of new node as head and previous as NULL */
    new_node->next = (*head_ref);
    new_node->prev = NULL;

    /* 4. change prev of head node to new node */
    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;

    /* 5. move the head to point to the new node */
    (*head_ref) = new_node;
}


/* Given a node as prev_node, insert a new node after the given node
*/
```

5

```
void insertAfter(Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL)
    {
        cout<<"the given previous node cannot be NULL";
        return;
    }

    /* 2. allocate new node */
    Node* new_node = new Node();

    /* 3. put in the data */
    new_node->data = new_data;

    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;

    /* 5. Make the next of prev_node as new_node */
    prev_node->next = new_node;

    /* 6. Make prev_node as previous of new_node */
    new_node->prev = prev_node;

    /* 7. Change previous of new_node's next node */
    if (new_node->next != NULL)
        new_node->next->prev = new_node;
}

/* Given a reference (pointer to pointer) to the head
of a DLL and an int, appends a new node at the end */

void append(Node** head_ref, int new_data)
{
    /* 1. allocate node */
    Node* new_node = new Node();

    Node* last = *head_ref; /* used in step 5*/

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. This new node is going to be the last node, so
        make next of it as NULL*/
    new_node->next = NULL;

    /* 4. If the Linked List is empty, then make the new
        node as head */
    if (*head_ref == NULL)
    {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }

    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;
    /* 6. Change the next of last node */
    last->next = new_node;

    /* 7. Make last node as previous of new node */
    new_node->prev = last;

    return;
}

// This function prints contents of
// linked list starting from the given node

void printList(Node* node)
{
    Node* last;
    cout<<"\nTraversal in forward direction \n";
    while (node != NULL)
    {
        cout<<" "<<node->data<<" ";
        last = node;
        node = node->next;
    }

    cout<<"\nTraversal in reverse direction \n";
    while (last != NULL)
    {
        cout<<" "<<last->data<<" ";
        last = last->prev;
    }
}


/* Driver program to test above functions*/

int main()
{
    /* Start with the empty list */

    Node* head = NULL;

    // Insert 6. So linked list becomes 6->NULL
    append(&head, 6);

    // Insert 7 at the beginning. So
    // linked list becomes 7->6->NULL
    push(&head, 7);

    // Insert 1 at the beginning. So
    // linked list becomes 1->7->6->NULL
    push(&head, 1);

    // Insert 4 at the end. So linked
    // list becomes 1->7->6->4->NULL
    append(&head, 4);



 // Insert 8, after 7. So linked
    // list becomes 1->7->8->6->4->NULL
```

	`insertAfter(head->next, 8);`
	
	`cout << "Created DLL is: ";`
	`printList(head);`
	
	`return 0;`
	`}`
	
	The expected output of the written code is___________ (Type your answer with appropriate space and escape sequence)__________________.		

Q36	void Sort(int a[], int n)
	
	{
	
	int i, j, min, temp;
	
	for (i = 0; i < n - 1; i++)
	
	{
	
	min = i;
	
	for (j = i + 1; j < n; j++)
	
	if (a[j] < a[min])
	
	min = j;
	
	temp = a[i];
	
	a[i] = a[min];
	
	a[min] = temp;
	
	}
	
	}
	
	The aforementioned code is designed for sorting the data using:
	
	(a) Quick Sort
	(b) Merge Sort
	(c) Selection Sort
	(d) Bubble Sort
		2	

Q37	Quick Sort is preferred for Arrays and Merge Sort for Linked List **(True/False)**	**2**	

Q38

A number is lucky if all digits of the number are different!
Written below is the code to check whether the entered no. Is lucky or not**. Fill in the blanks to let the code execute correct result;**

```
#include<iostream>
using namespace std;

// This function returns true if n is lucky

bool isLucky(int n)
{
    bool arr[10];
    for (int i=0; i<10; i++)
        arr[i] = (true or false?); // Select one out of them

    // Traverse through all digits of given number
    while (n > 0)
    {
        // Find the last digit
        int digit = n%10;

        if (arr[digit])
           return (true or false ?); // Select one out of them

        arr[digit] = (true or false ?); // Select one out of them

                n = n/10;
    }
    return (true or false ?); // Select one out of them
}

// Driver program to test above function.

int main()
{
    int arr[] = {1291, 897, 4566, 1232, 80, 700};
    int n = sizeof(arr)/sizeof(arr[0]);

    for (int i=0; i<n; i++)
        isLucky(arr[i])? cout << arr[i] << " is Lucky \n":
                         cout << arr[i] << " is not Lucky \n";
    return 0;
}
```

4

Q39

Which out of the following is not an access specifier in C++:

(a) Public
(b) Private
(c) Protected
(d) Default

2

Q40	Bubble Sort is so named because it bubbles the smallest element to the middle of the array. (True/False)	**2**	
Q41	______________________Sort method is optimal because the sorted array is developed without using any extra storage space	**2**	
Q42	The Sequential Search method on sorted lists is faster than the indexed method. (True/False)	**1**	
Q43	When an element needs to be removed from the stack, the pop operation is performed. (True/ False)	**1**	
Q44	Data members for any class are initialized using the following: (a) Destructor (b) Friend function (c) Constructor (d) Array (e) None of these	**2**	
Q45	Loop statement to be used when a user want to execute a task at least once even if the condition set for the loop is false. (a) while loop (b) do while loop (c) for loop (d) All of them	**2**	