Name: Enrolment No:	UPES

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

End Semester Examination, December 2021

Programme Name: B.Tech. ECE **Semester : III**
Course Name : Advanced Programming **Time : 03 hrs**
Course Code : ECEG2038 **Max. Marks: 100**
Nos. of page(s) : 15

Instructions:

1. **Attempt all the questions (Theory, Numerical, Case study etc.)**
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**

SECTION A [20 marks]

S. No.		Marks	CO

Q 1.
 	
(I)
	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];
		2+2	**CO2**

	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		

(II)	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		

Q 2.	
	
	`#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();`
		4	**CO3**

```
    /* 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
*/

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)_____.

Q 3.	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;	**4**	**CO1**

```
                                                                ^
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;


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 4. **(I)** **(II)**	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 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+2**	**CO4**

Q5.	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]`
	`  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) None of these
	(d) Adding 100 to the upper diagonal elements and subtracting 100 from diagonal elements of A
		4	**CO2**

SECTION B [40 marks]			

Q 6.	Implement a Doubly Linked List via writing a code in C++ as per the instruction(s) dictated as hereunder:
	
	#include<iostream>
	using namespace std;
	
	struct Node
	{
	int data;
	struct Node* next;
	struct Node* prev;
	};
	
	_____// global variable - pointer to head node.
	
	_____// Create a new Node and returns pointer to it.
	
	_____//Inserts a Node at head of doubly linked list
	
	_____//Inserts a Node at tail of Doubly linked list
	
	_____//Prints all the elements in linked list in forward traversal order
	
	_____/*Driver code to test the implementation*/
	
	_____// empty list. Set head as NULL.
	
	_____// Calling an Insert and printing list both in forward as well as reverse direction.
		10	**CO3**

Q 7. **(I)**	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**	**5+5**	**CO2**

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;
  }
}
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;
	}		

II	**In which memory a String is stored, when we create a string using new operator? Justify your answer with suitable example.**		
Q 8.	Brief about the following terms: (a) Selection sort (b) Quick Sort (c) Pointers with function (d) EnQueue & DeQueue (illustrate along with code) (e) Insertion Sort	**10**	**CO4**
Q9. **(I)**	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;**	**5+5**	**CO1**

```
#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;
}
```

(II)

(A)

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		
(B)	Bubble Sort is so named because it bubbles the smallest element to the middle of the array. (True/False)		
(C)	_____Sort method is optimal because the sorted array is developed without using any extra storage space		
SECTION-C [40 marks]			
Q 10.			
(I)	**Indicate the following with the help of its appropriate Syntax (in C++):** (a) Dynamic memory allocation (b) Self-referencing structure (c) Function passing pointers (d) Recursive function	**8**	**CO3**
(II)	Enlist the advantages of Linked list and real world examples of Stack implementation in the domain of Data Structure;	**7**	**CO3**
(III)	Write a C++ code to implement a Queue using linked list.	**5**	**CO3**
Q 11. **(I)** **(A)**	Which of the following scenario is true for the statement - "**Arrays are best data structures"?** (a) For the size of the structure and the data in the structure are constantly changing (b) For relatively permanent collections of data (c) both a and b (d) none of the above	**3+3**	**CO4**
(B)	What will be the final elements on the stack if the following sequence of operations is executed? * Push(a,s); * Push(b,s); * Pop(s); * Push(c,s);		

	Where a, b, c are the data elements and s is the stack. (i) abc (ii) ac (iii) acb (iv) b		
(II)	Write a snippet code to implement Singly Linked list using Self Referencing Structure.	**7**	
(III)	**Match the following:** **1. Linked List — (a) CPU Scheduling** **2. Bubble Sort — (b) Sparse Matrix representation** **3. Queue — (c) Delimiter Checking** **4. Stack — (d) implement undo-redo feature** **5. Array — (e) Global variables storage** **6. Doubly linked list — (f) Book search in Library** **7. Heap memory — (g) implementation of pointers**	**7**	