LAB MSE

1. Design, Develop and Implement a menu driven Program in C for the following

a) To create an Array of N Integer Elements and store n values.

•Inserting an Element (ELEM) at a given valid Position (POS)

• Deleting an Element at a given valid Position POS)

•Display of Array Elements

•Exit.

Support the program with functions for each of the above operations.

b) To create student structure with fields Roll No, Name, Semester, marks in 3 subjects. And Write functions to

•Enter 5 students' details and display the same using pointer to structure

•Find Student wise and subject wise total marks and display the same.

A)

```
#include <stdio.h>
#include <stdlib.h>

int a[100];
int pos, elem;
int n = 0;

void create();
void display();
void insert();
void del();

int main() {
    int choice;
    while (1) {
        printf("\n\n~~~~~MENU~~~~~");
        printf("\n->1: Create an array of n integers");
        printf("\n->2: Display the array elements");
        printf("\n->3: Insert an element at a given position");
        printf("\n->4: Delete an element at a given position");
        printf("\n->5: Exit");
```

```
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                create();
                break;
            case 2:
                display();
                break;
            case 3:
                insert();
                break;
            case 4:
                del();
                break;
            case 5:
                exit(0);
                break;
            default:
                printf("\nInvalid choice. Please enter a valid option.");
        }
    }
    return 0;
}

void create() {
    int i;
    printf("\nEnter the number of elements: ");
    scanf("%d", &n);
    if (n > 100) {
        printf("Error: Array size cannot exceed 100.\n");
        n = 0;
        return;
    }
    printf("Enter the %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
}

void display() {
    int i;
    if (n == 0) {
        printf("\nThe array is empty.");
        return;
    }
    printf("\nThe array elements are: ");
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
}

void insert() {
    int i;
    if (n == 100) {
        printf("\nArray is full. Insertion not possible.");
        return;
    }

    do {
        printf("\nEnter a valid position to insert the element (0 to %d): ", n);
        scanf("%d", &pos);
    } while (pos > n || pos < 0);

    printf("\nEnter the value to be inserted: ");
    scanf("%d", &elem);

    for (i = n - 1; i >= pos; i--) {
        a[i + 1] = a[i];
    }

    a[pos] = elem;
    n = n + 1;

    printf("\nElement inserted successfully.");
    display();
}

void del() {
    int i;
    if (n == 0) {
        printf("\nArray is empty. Deletion not possible.");
        return;
    }

    do {
        printf("\nEnter a valid position to delete the element (0 to %d): ", n - 1);
        scanf("%d", &pos);
    } while (pos >= n || pos < 0);

    elem = a[pos];
    printf("\nDeleted element is: %d", elem);

    for (i = pos; i < n - 1; i++) {
        a[i] = a[i + 1];
    }

    n = n - 1;

    display();
}
```

B)

```
#include <stdio.h>

struct student {
    int rollno;
    char name[50];
    int sem;
    float mark1, mark2, mark3;
};

int main() {
    struct student s[5], *ptr;
    int i;
    float total, s1total = 0, s2total = 0, s3total = 0;

    ptr = s;

    // Input student details
    for (i = 0; i < 5; i++) {
        printf("\nEnter details of student %d:\n", i + 1);
        printf("Enter name: ");
        scanf("%s", ptr->name);
        printf("Enter roll number: ");
        scanf("%d", &ptr->rollno);
        printf("Enter semester: ");
        scanf("%d", &ptr->sem);
        printf("Enter marks of subject 1: ");
        scanf("%f", &ptr->mark1);
        printf("Enter marks of subject 2: ");
        scanf("%f", &ptr->mark2);
        printf("Enter marks of subject 3: ");
        scanf("%f", &ptr->mark3);
        ptr++;
    }

    // Reset pointer
    ptr = s;
    // Display student details + total
    printf("\n--- Student Details and Totals ---\n");
    for (i = 0; i < 5; i++) {
        total = ptr->mark1 + ptr->mark2 + ptr->mark3;
        printf("\nName: %s, Roll No: %d, Semester: %d", ptr->name, ptr->rollno, ptr->sem);
        printf("\nMarks: %.2f, %.2f, %.2f", ptr->mark1, ptr->mark2, ptr->mark3);
        printf("\nTotal marks of student %d = %.2f\n", i + 1, total);
        ptr++;
    }

    // Reset pointer
    ptr = s;

    // Calculate subject-wise totals
    for (i = 0; i < 5; i++) {
        s1total += ptr->mark1;
        s2total += ptr->mark2;
        s3total += ptr->mark3;
        ptr++;
    }

    printf("\n--- Subject-wise Totals ---\n");
    printf("Subject 1 total = %.2f\n", s1total);
    printf("Subject 2 total = %.2f\n", s2total);
    printf("Subject 3 total = %.2f\n", s3total);

    return 0;
}
```

2. Design, Develop and Implement a menu driven Program in C for the following operations on

a) STACK of Integers (Array with structure Implementation of Stack with size MAX)

- Push an Element on to Stack.
- Pop an Element from Stack.
- Demonstrate Overflow and Underflow situations on Stack.
- Display the status of Stack .
- Exit

```
#include <stdio.h>
#include <stdlib.h>

#define MAXSIZE 10

int top = -1;

void push(int a[], int item)
{
   top = top + 1;
   a[top] = item;
}

int pop(int a[])
{
   int item;
   item = a[top];
   top = top - 1;
   return item;
}

void display(int a[])
{
   int i;
   if (top == -1)
   {
      printf("the stack is empty");
   }
   else
   {
        printf("the stack elements are ");
    }
    for (i = top; i >= 0; i--)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
}

int main()
{
    int a[10], choice, item;
    while (1)
    {
        printf("Enter the choice \n");
        printf("\n 1.push \n 2.pop \n 3.Display \n 4.Exit \n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            if (top == MAXSIZE - 1)
            {
                printf("the stack is full\n");
            }
            else
            {
                printf("enter the element to be pushed: ");
                scanf("%d", &item);
                push(a, item);
            }
            break;

        case 2:
            if (top == -1)
            {
                printf("the stack is empty\n");
            }
            else
            {
                item = pop(a);
                printf("Popped element: %d\n", item);
            }
            break;

        case 3:
            display(a);
            break;
        case 4:
            exit(0);
        }
    }
    return 0;
}
```

B)

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

// Defines the structure for the stack
struct stack {
    int top;
    int items[MAX];
};

// Function prototypes
void push(int, struct stack *);
void pop(struct stack *);
void display(struct stack *);

int main() {
    struct stack s;
    s.top = -1; // Initialize the top of the stack
    int choice, item;

    // Infinite loop for the menu
    for (;;) {
        printf("\nEnter your choice:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the item to push: ");
                scanf("%d", &item);
                push(item, &s);
                break;
            case 2:
                pop(&s);
                break;
            case 3:
                display(&s);
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}

// Function to add an item to the stack
void push(int item, struct stack *s) {
    if (s->top == MAX - 1) {
        printf("Stack is full. Cannot push item.\n");
    } else {
        s->top++;
        s->items[s->top] = item;
    }
}

// Function to remove an item from the stack
void pop(struct stack *s) {
    int item;
    if (s->top == -1) {
        printf("Stack is empty. Cannot pop item.\n");
    } else {
        item = s->items[s->top];
        s->top--;
        printf("%d deleted from the stack.\n", item);
    }
}

// Function to display the elements of the stack
void display(struct stack *s) {
    int t = s->top;
    if (s->top == -1) {
        printf("Stack is empty.\n");
    } else {
        printf("Elements in the stack are: ");
        while (t > -1) {
            printf("%d ", s->items[t--]);
        }
        printf("\n");
    }
}
```

3. Queue of Integers (Array with structure Implementation of Queue with size MAX) • Insert an Element in to Queue.

- Delete an Element from Queue.
- Demonstrate Overflow and Underflow situations on Queue. • Display the status of Queue .
- Exit

Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#define MAX 100

int queue[MAX];
int front = 0;
int rear = -1;

int isEmpty() {
    return rear < front;
}

int isFull() {
    return rear == MAX - 1;
}

void enqueue(int value) {
    if (isFull()) {
        printf("Queue is full\n");
        return;
    }
    rear++;
    queue[rear] = value;
    printf("Enqueued: %d\n", value);
}

int dequeue() {
    if (isEmpty()) {
        printf("Queue is empty\n");
        return -1;
    }
    int val = queue[front];
    front++;
    return val;
}

void display() {
    if (isEmpty()) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int choice, value;
    while (1) {
        printf("\n--- Queue Menu ---\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                value = dequeue();
                if (value != -1)
                    printf("Dequeued: %d\n", value);
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
}
```

B)

```
#include <stdio.h>

#define MAX 100

struct Queue {
    int queue[MAX];
    int front;
    int rear;
};

struct Queue q = { .front = 0, .rear = -1 };

void enqueue(int value) {
    if (q.rear == MAX - 1) {
        printf("Queue is full\n");
        return;
    }
    q.rear++;
    q.queue[q.rear] = value;
    printf("Enqueued: %d\n", value);
}

void dequeue() {
    if (q.front > q.rear) {
        printf("Queue is empty\n");
        return;
    }
    int val = q.queue[q.front];
    q.front++;
    printf("Deleted item is: %d\n", val);
}

void display() {
    if (q.front > q.rear) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    for (int i = q.front; i <= q.rear; i++) {
        printf("%d ", q.queue[i]);
    }
    printf("\n");
}
```

```
int main() {
    int choice, value;

    while (1) {
        printf("\n--- Queue Menu ---\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
}
```

4. Design, Develop and Implement a Program in C for the following Stack Applications

    - Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^ b
    - Solving Tower of Hanoi problem with n disks

A)

```
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <stdlib.h> // Required for exit()

#define MAX 100

struct stack
{
   int top;
   double a[MAX];
};

void push(char, struct stack *);
double pop(struct stack *);
double compute(double, double, char);

int main()
{
   struct stack s;
   s.top = -1;
   int i;
   double op1, op2, res;
   char postfix[100], symb;

   printf("Enter postfix expression: ");
   gets(postfix);

   for (i = 0; postfix[i] != '\0'; i++)
   {
      symb = postfix[i];
      if (isdigit(symb))
      {
         push(symb, &s);
      }
      else
      {
            op2 = pop(&s);
            op1 = pop(&s);
            res = compute(op1, op2, symb);
            s.a[++s.top] = res;
        }
    }
    res = pop(&s);
    printf("Result after evaluation = %f\n", res);
    return 0;
}

void push(char symb, struct stack *s)
{
    if (s->top == MAX - 1)
    {
        printf("Stack Overflow\n");
    }
    else
    {
        s->a[++s->top] = symb - '0';
    }
}

double pop(struct stack *s)
{
    if (s->top == -1)
    {
        printf("Stack Underflow or Invalid Expression\n");
        exit(1);
    }
    else
    {
        // Corrected: Now returns the value from the stack
        return (s->a[s->top--]);
    }
}

double compute(double op1, double op2, char symb)
{
    switch (symb)
    {
    case '+':
        return (op1 + op2);
    case '-':
        return (op1 - op2);
    case '*':
        return (op1 * op2);
    case '/':
        return (op1 / op2);
    case '$':
    case '^':
        return (pow(op1, op2));
    default:
        printf("Invalid operator '%c'\n", symb);
        exit(1);
    }
}
```

B)

```
#include <stdio.h>
#include <stdlib.h>

// Function prototype for the tower function
void tower(int n, char sp, char ap, char dp);

int main()
{
    int n;
    printf("enter the number of discs: ");
    scanf("%d", &n);
    printf("The discs movements are:\n");
    tower(n, 'A', 'C', 'B');
    return 0;
}

void tower(int n, char sp, char ap, char dp)
{
    if (n == 1)
    {
        printf("\nMoving disc %d from %c to %c", n, sp, dp);
        return;
    }

    // Move n-1 discs from source (sp) to auxiliary (ap) using destination (dp) as auxiliary.
    tower(n - 1, sp, dp, ap);

    printf("\nMove disc %d from %c to %c", n, sp, dp);

    // Move n-1 discs from auxiliary (ap) to destination (dp) using source (sp) as auxiliary.
    tower(n - 1, ap, sp, dp);
}
```

5.Design develop and implement a program in c for converting an infix expression to postfix expression program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands

```
#include <stdio.h>
#include <stdlib.h> // Required for exit()
#include <ctype.h>  // Required for isalpha()

#define max 100
#define TRUE 1
#define FALSE 0

struct stack
{
   int top;
   char items[max];
};

struct stack s;
char infix[max], postfix[max];
int pos = 0;

void convert();
void push(char);
char pop();
int precedence(char);
int f = 0;
int empty();
int stackfull();

int main()
{
   s.top = -1;
   printf("Enter the infix expression\n");
   gets(infix);
   convert();
```

```
    if (f == 0)
    {
        printf("The postfix expression is\n");
        puts(postfix);
    }
    return 0;
}

void convert()
{
    if (infix[0] == '\0')
    {
        f = 1;
        printf("Invalid input\n");
        return;
    }
    int i;
    char symbol, temp;
    for (i = 0; infix[i] != '\0'; i++)
    {
        symbol = infix[i];
        switch (symbol)
        {
        case '(':
            push(symbol);
            break;
        case ')':
            while ((temp = pop()) != '(')
                postfix[pos++] = temp;
            break;
        case '+':
        case '-':
        case '*':
        case '/':
        case '$':
            while (!empty() && precedence(s.items[s.top]) >= precedence(symbol) &&
precedence(symbol) != -1)
            {
                temp = pop();
                postfix[pos++] = temp;
            }
            push(symbol);
            break;
        default:
            if (!isalpha(symbol))
            {
                printf("Invalid input\n");
                f = 1;
                return;
            }
            else
            {
                postfix[pos++] = symbol;
                break;
            }
        }
    }
    while (!empty())
    {
        temp = pop();
        postfix[pos++] = temp;
    }
    postfix[pos] = '\0'; // Null-terminate the string for puts()
}

void push(char ele)
{
    if (stackfull())
        printf("Stack is full\n");
    else
        s.items[++s.top] = ele;
}

char pop()
{
    if (empty())
    {
        printf("Stack is empty\n");
        exit(0);
    }
    else
        return (s.items[s.top--]);
}

int stackfull()
{
    if (s.top == max - 1)
        return TRUE;
    else
        return FALSE;
}

int empty()
{
    if (s.top == -1)
        return TRUE;
    else
        return FALSE;
}

int precedence(char symbol)
{
    switch (symbol)
    {
    case '$':
        return 3;
    case '*':
    case '/':
        return 2;
    case '+':
    case '-':
        return 1;
    case '(':
    case ')':
        return (0);
    default:
        printf("Invalid input\n");
        return -1;
    }
}
```

6. Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

    - Insert an Element on to Circular QUEUE.
    - Delete an Element from Circular QUEUE.
    - Demonstrate Overflow and Underflow situations on Circular QUEUE
      d. Display the status of Circular QUEUE.
    - Exit

    Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include <stdlib.h>
#define maxsize 5
int queue[maxsize], front = 0, rear = -1, count = 0;
void insert(int item) {
   if (count == maxsize) {
      printf("Queue overflow");
      return;
   }
   rear = (rear + 1) % maxsize;
   queue[rear] = item;
   count++;
}
void delete() {
   int item;
   if (count == 0) {
      printf("Queue underflow");
   } else {
      item = queue[front];
      printf(" The value deleted is: %d", item);
      front = (front + 1) % maxsize;
      count--;
   }
}
void display() {
   int i, temp_front = front;

   if (count == 0) {
        printf("Queue underflow");
    } else {
        for (i = 1; i <= count; i++) {
            printf("%d\t", queue[front]);
            front = (front + 1) % maxsize;
        }
    }
}
int main() {
    int value, choice;
    while (1) {
        printf("\n-MENU-----\n");
        printf(" 1. Insertion\n 2. Deletion\n 3. Display\n4. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the value to be inserted: ");
                scanf("%d", &value);
                insert(value);
                break;
            case 2:
                delete();
                break;
            case 3:
                printf("Queue elements: ");
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("Enter valid choice");
                return 0;
        }
    }
}   }
}
```

7. Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) for data of integers.

- Create a SLL of Data using front insertion.
- Display the status of SLL and count the number of nodes in it. • Demonstration of stack
- Demonstration of Queue. • Exit

```
#include <stdio.h>
#include <stdlib.h>

struct node {
   int info;
   struct node *link;
};

typedef struct node *NODE;

NODE getnode() {
   NODE x;
   x = (NODE)malloc(sizeof(struct node));
   if (x == NULL) {
      printf("Memory not available");
      return x;
   }
   return x;
}

void freenode(NODE x) {
   free(x);
}

NODE insert_front(NODE first, int item) {
   NODE temp;
   temp = getnode();
   temp->info = item;
   temp->link = first;
   return (temp);
}

NODE insert_rear(int item, NODE first) {
   NODE temp, cur;
   temp = getnode();
   temp->info = item;
   temp->link = NULL;
   if (first == NULL)
      return temp;
   cur = first;
   while (cur->link != NULL) {
      cur = cur->link;
   }
   cur->link = temp;
   return first;
}

NODE delete_front(NODE first) {
   NODE temp;
   if (first == NULL) {
      printf("List is empty\n");
      return first;
   }
   temp = first;
   temp = temp->link;
   printf("Deleted data is %d\n", first->info);
   freenode(first);
   return temp;
}

NODE display_front(NODE first) {
   NODE temp;
   int count = 0;
   if (first == NULL) {
      printf("List is empty\n");
      return first;
   }
   temp = first;
   while (temp != NULL) {
      printf("%d\t", temp->info);
      temp = temp->link;
      count++;
   }
   printf("\n");
   printf("Number of Nodes : %d", count);
}
```

```
int main() {
    NODE first = NULL;
    int option, value, choice;
    while (1) {
        printf("\n   MENU-----\n");
        printf("1. Insert front\n 2. Display\n 3. Stack\n4. Queue\n 5. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the value to be inserted: ");
                scanf("%d", &value);
                first = insert_front(first, value);
                break;
            case 2:
                printf("Queue elements: ");
                display_front(first);
                break;
            case 3:
                option = 0;
                while (option != 4) {
                    printf("\n MENU STACK \n");
                    printf("1. Insert front\n 2. Delete front\n 3. Display\n 4. Exit");
                    printf("\nEnter your choice:");
                    scanf("%d", &option);
                    switch (option) {
                        case 1:
                            printf("Enter the value to be inserted: ");
                            scanf("%d", &value);
                            first = insert_front(first, value);
                            break;
                        case 2:
                            first = delete_front(first);
                            break;
                        case 3:
                            printf("Queue elements: ");
                            display_front(first);
                            break;
                        case 4:
                            break;
                        default:
                            printf("Enter valid choice");
                    }
                }
                break;
            case 4:
                option = 0;
                while (option != 4) {
                    printf("\n MENU QUEUE \n");
                    printf(" 1. Insert rear\n 2. Delete front\n 3. Display\n 4. Exit");
                    printf("\nEnter your choice: ");
                    scanf("%d", &option);
                    switch (option) {
                        case 1:
                            printf("Enter the value to be inserted: ");
                            scanf("%d", &value);
                            first = insert_rear(value, first);
                            break;
                        case 2:
                            first = delete_front(first);
                            break;
                        case 3:
                            printf(" Queue elements: ");
                            display_front(first);
                            break;
                        case 4:
                            break;
                        default:
                            printf("Enter valid choice");
                    }
                }
                break;
            case 5:
                exit(0);
            default:
                printf("Enter valid choice");
        }
    }
    return 0;
}
```