1. **Develop a Program in C for the following:**

a) **Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).**

b) **Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.**

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

#define NO_OF_DAYS 7

typedef struct
{
   char *name_of_day;
   int  date_of_day;
   char *activity_descr;
} CALENDER;

void create_calender(CALENDER a[], int i, char name[], int date, char activity[])
{
   a[i].name_of_day = (char *) malloc(strlen(name) + 1);

   strcpy(a[i].name_of_day, name);

   a[i].date_of_day = date;

   a[i].activity_descr = (char *) malloc(strlen(activity) + 1);
   strcpy(a[i].activity_descr, activity);
}


void read_calender(CALENDER a[])
{
   int i, date;
   char name[10], activity[10];

   for (i = 0; i < NO_OF_DAYS; i++)
```

```c
    {
        scanf("%s", name);
        scanf("%d", &date);
        scanf("%s", activity);

        create_calender(a, i, name, date, activity);
    }
}
void print_weeks_activity(CALENDER a[])
{
    int i;

    printf("Weeks activity\n");
    for (i = 0; i < NO_OF_DAYS; i++)
    {
        printf("%-10s : %s\n", a[i].name_of_day, a[i].activity_descr);
    }
}

void main()
{
    CALENDER a[NO_OF_DAYS];

    printf("Name  Date Activity\n");
    read_calender(a);

    print_weeks_activity(a);
}
```

Output:

```
Name   Date Activity
Monday      1     Working
Tuesday     2     Gym
Wednesday 3      Studying
Thursday    4     Meeting
Friday      5     Shopping
Saturday    6     Cleaning
Sunday      7     Relaxing
Weeks activity
Monday      : Working
Tuesday     : Gym
Wednesday   : Studying
Thursday    : Meeting
Friday      : Shopping
Saturday    : Cleaning
Sunday      : Relaxing


...Program finished with exit code 0
Press ENTER to exit console.
```

3. Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

a. Push an Element on to Stack

b. Pop an Element from Stack

c. Demonstrate how Stack can be used to check Palindrome

d. Demonstrate Overflow and Underflow situations on Stack

e. Display the status of Stack

f. Exit Support the program with appropriate functions for each of the above operations

```c
#include <stdio.h>
#include <stdlib.h>
#define STACK_SIZE 5
int top = -1;
int stack[10];
void push(int item)
{
if (top == STACK_SIZE - 1)
{
printf("Overflow of stack\n");
return;
}
top++;
stack[top] = item;
}
void pop()
{
if (top == -1)
{
printf("Stack underflow\n");
return;
}
printf("Item deleted = %d\n", stack[top]);
top = top - 1;
}

void display()
{
    int i;
```

```c
    if (top == -1)
    {
        printf("Stack is empty\n");
        return;
    }

    printf("Stack: ");
    for (i = 0; i <= top; i++)
        printf("%d ", stack[i]);

    printf("\n");
}

void palindrome(char str[])
{
    int i;

    for(i = 0; str[i] != '\0'; i++)
    {
        stack[++top] = str[i];
    }

    for(i = 0; str[i] != '\0'; i++)
    {
        if(str[i] == stack[top--])
            continue;
        printf("%s: Not a palindrome\n",str);
        return;
    }

        printf("%s: Palindrome\n",str);
}

int main()
{
    int item, choice;
    char str[10];

    for (;;)
    {
        printf("1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                printf("Enter the item: ");
```

```c
            scanf("%d", &item);
            push(item);
            break;

        case 2:
            pop();
            break;

        case 3:
            display();
            break;

        case 4:
            printf("Enter the string: ");
            scanf(" %[^\n]", str);
            palindrome(str);
            break;

        default:
            exit(0);
        }
    }
}
```

Output:

```
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 1
Enter the item: 10
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 1
Enter the item: 20
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 1
Enter the item: 30
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 1
Enter the item: 40
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 1
Enter the item: 50
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 1
Enter the item: 60
Overflow of stack
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 3
Stack: 10 20 30 40 50
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 2
Item deleted = 50
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 2
Item deleted = 40
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 2
Item deleted = 30
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 2
Item deleted = 20
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 2
Item deleted = 10
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 2
Stack underflow
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 4
Enter the string: level
level: Palindrome
1: Insert 2: Delete 3: Display 4:Palindrome 5: Exit : 4
Enter the string: rare
rare: Not a palindrome
```

4. Develop 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.

```c
#include <stdio.h>
#include <string.h>

int F (char symbol)
{
    switch(symbol)
    {
        case '#' : return -1;

        case '+' :
        case '-' : return 2;

        case '*' :
        case '/' :
        case '%' : return 4;

        case '^' :
        case '$' : return 5;

        case '(' : return 0;

        default  : return 8;
    }
}

int G (char symbol)
{
    switch(symbol)
    {
        case ')' : return 0;

        case '+' :
        case '-' : return 1;

        case '*' :
        case '/' :
        case '%' : return 3;
```

```c
        case '^' :
        case '$' : return 6;

        case '(' : return 7;

        default  : return 9;
    }
}
void infix_2_postfix(char infix[], char postfix[])
{
    int    i, j = 0, top = -1;
    char   s[20];

    s[++top] = '#';
    for (i = 0; infix[i] != '\0'; i++)
    {
        while ( F( s[top] ) > G( infix[i] ) )
            postfix[j++] = s[top--];

        if ( F( s[top] ) != G( infix[i] ) )
            s[++top] = infix[i];
        else
            s[top--];
    }

    while (s[top] != '#')
    {
        postfix[j++] = s[top--];
    }

    postfix[j] = '\0';
}

void main()
{
    char infix[20], postfix[20];

    printf("Infix :");
    scanf("%[^\n]", infix);

    infix_2_postfix(infix, postfix);

    printf("Postfix:");
    printf("%s\n", postfix);
}
```

Output:

```
Infix :(a+b)+c/d*e
Postfix:ab+cd/e*+
```

5. Develop a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

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

double compute(double operand1, char operator, double operand2) {
    switch (operator) {
        case '+': return operand1 + operand2;
        case '-': return operand1 - operand2;
        case '*': return operand1 * operand2;
        case '/': return operand1 / operand2;
        case '%': return fmod(operand1, operand2);
        case '^': return pow(operand1, operand2);
        default:
            exit(0);
    }
}

double evaluate(char postfix[]) {
    int i, top = -1;
    double stack[20], operand1, operand2;

    for (i = 0; postfix[i] != '\0'; i++) {
        if (postfix[i] >= '0' && postfix[i] <= '9') {
            stack[++top] = postfix[i] - '0';
        } else {
            operand2 = stack[top--];
            operand1 = stack[top--];
            stack[++top] = compute(operand1, postfix[i], operand2);
        }
    }

    return stack[top--];
}

int main() {
    char postfix[20];
    double result;

    printf("Enter Postfix expression: ");
    scanf("%s", postfix);
```
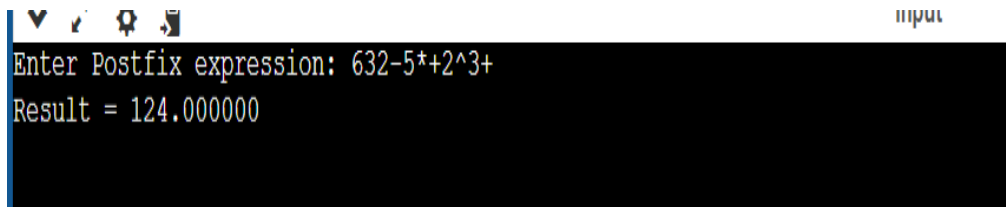
```
    result = evaluate(postfix);

    printf("Result = %lf\n", result);

    return 0;
}
```

Output:


```
Enter Postfix expression: 632-5*+2^3+
Result = 124.000000
```

b. Solving Tower of Hanoi problem with n disks.

```
#include <stdio.h>

void Transfer(int n, char source, char temp, char dest)
{
    if (n == 0) return;

    Transfer(n - 1, source, dest, temp);

    printf("Move disk %d from %c to %c\n", n, source, dest);

    Transfer(n - 1, temp, source, dest);
}

void main()
{
    int n;

    printf("Enter number of disks : ");
    scanf("%d", &n);

    Transfer(n, 'A', 'B', 'C');
}
```

Output:

```
Enter number of disks : 3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C


...Program finished with exit code 0
Press ENTER to exit console.
```

6. Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
 d. Display the status of Circular QUEUE
 e. Exit Support the program with appropriate functions for each of the above operations.

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int queue[10], front = 0, rear = -1, count = 0;
void insert_rear(int item)
{
if (count == MAX)
{
printf("Q overflow\n");
return;
}
rear = (rear + 1) %MAX;
queue[rear] = item;
count++;
}
void delete_front()
{
if (count == 0)
{
printf("Q underFlow\n");
return;
}
printf("ITem deleted = %d\n", queue[front]);
front = (front + 1) % MAX;
count--;
}
void display()
```

```c
{
int i, temp;

    if (count == 0)
    {
       printf("Q is empty\n");
       return;
    }

    printf("Queue : ");
    temp = front;
    for (i = 1; i <= count; i++)
    {
       printf("%d ", queue[temp]);
       temp = (temp + 1) % MAX;
    }

    printf("\n");
}

void main()
{
    int item, choice;

    for(;;)
    {
       printf("1: insert 2:Delete 3:Display 4:Exit : ");
       scanf("%d", &choice);
       switch(choice)
       {
          case 1:
                printf("Enter the item : ");
                scanf("%d", &item);
                insert_rear(item);
                break;

          case 2:
                delete_front();
                break;

          case 3:
                display();
                break;

          default:
                exit(0);
       }
```

```
    }
}
```

Output:

```
1: insert 2:Delete 3:Display 4:Exit : 1
Enter the item : 10
1: insert 2:Delete 3:Display 4:Exit : 1
Enter the item : 20
1: insert 2:Delete 3:Display 4:Exit : 1
Enter the item : 30
1: insert 2:Delete 3:Display 4:Exit : 1
Enter the item : 40
1: insert 2:Delete 3:Display 4:Exit : 1
Enter the item : 50
1: insert 2:Delete 3:Display 4:Exit : 1
Enter the item : 60
Q overflow
1: insert 2:Delete 3:Display 4:Exit : 3
Queue : 10 20 30 40 50
1: insert 2:Delete 3:Display 4:Exit : 2
ITem deleted = 10
1: insert 2:Delete 3:Display 4:Exit : 2
ITem deleted = 20
1: insert 2:Delete 3:Display 4:Exit : 2
ITem deleted = 30
1: insert 2:Delete 3:Display 4:Exit : 2
ITem deleted = 40
1: insert 2:Delete 3:Display 4:Exit : 2
ITem deleted = 50
1: insert 2:Delete 3:Display 4:Exit : 2
Q underFlow
1: insert 2:Delete 3:Display 4:Exit : 4


...Program finished with exit code 0
Press ENTER to exit console.
```

7. Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo a. Create a SLL of N Students Data by using front insertion. b. Display the status of SLL and count the number of nodes in it c. Perform Insertion / Deletion at End of SLL d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack) e. Exit

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

struct node
{
    char    usn[25];
    char    name[25];
    char    programme[25];
    char    phone[12];
    int     sem;
    struct node*link;
};
```

```c
typedef struct node *NODE;

NODE getnode()
{
   NODE x;

   x = (NODE) malloc (sizeof(struct node));

   if(x == NULL)
   {
      printf("not enough memory");
      exit(0);
   }

   return x;
}

NODE insert_front(char usn[], char name[], char programme[],char phone[], int sem, NODE first)
{
   NODE temp;

   temp = getnode();

   strcpy(temp ->usn, usn);
   strcpy(temp ->name,name);
   strcpy(temp ->programme, programme);
   strcpy(temp ->phone, phone);
   temp -> sem = sem;

   temp -> link = first;

   return temp;
}

NODE insert_rear(char usn[], char name[], char programme[], char phone[], int sem, NODE first)
{
   NODE temp, cur;

   temp = getnode();

   strcpy(temp ->usn, usn);
   strcpy(temp ->name,name);
   strcpy(temp ->programme, programme);
   strcpy(temp ->phone, phone);
   temp -> sem = sem;
```

```c
        temp -> link = NULL;

        if (first == NULL) return temp;

        cur = first;
        while(cur -> link != NULL)
        {
            cur=cur -> link;
        }

        cur -> link = temp;

        return first;

}

NODE delete_rear(NODE first)
{
    NODE prev,cur;

    if(first == NULL)
    {
        printf("Student list is empty\n");

        return NULL;
    }

    if(first -> link == NULL)
    {
        printf("The student with usn %s is deleted\n " ,first ->usn);
        free(first);

        return NULL;
    }

    cur = first;
    while(cur -> link != NULL)
    {
        prev = cur;
        cur = cur -> link;
    }

    prev -> link = NULL;

    printf("The student with usn %s is deleted\n", cur -> usn);
    free(cur);
```

```c
      return first;
}

NODE delete_front(NODE first)
{
   NODE prev, cur;

   if( first == NULL )
   {
      printf("students list is empty\n");

      return NULL;
   }

   if(first -> link == NULL)
   {
      printf("The student with usn %s is deleted\n", first -> usn);
      free(first);

      return NULL;
   }


   cur = first -> link;

   printf(" The student with usn %s is deleted\n", first -> usn);
   free(first);

   return cur;
}

void display(NODE first)
{
   NODE cur;

   printf("Student list\n");

   printf("usn    name   programme   phone   sem");

   if (first == NULL)
   {
      printf( "empty");

      return;
   }

   cur = first;
```

```c
    while(cur != NULL)
    {
        printf("\n%-13s", cur -> usn);
        printf("%-13s", cur -> name);
        printf("%-13s", cur -> programme);
        printf("%-13s", cur -> phone);
        printf("%-5d", cur -> sem);

        cur = cur -> link;

    }

    printf("\n");

}

void main()
{
    char usn[25], name[25],programme[25], phone[12];
    int choice, sem;
    NODE first;

    for(;;)
    {
        printf("\n\n 1:Front insert  2:rear insert\n");
        printf("3:Front delete  4:rear delete\n");
        printf("5: Display  6:exit\n");
        printf("Enter your choice  :");
        scanf("%d", &choice);

        switch(choice)
        {
            case 1:
                printf("USN: ");
                scanf("%s",usn);
                printf("Name: ");
                scanf("%s", name);
                printf("programme: ");
                scanf("%s", programme);
                printf("phone: ");
                scanf("%s", phone);
                printf("sem: ");
                scanf("%d", &sem);

                first = insert_front(usn, name, programme, phone, sem, first);
                break;
```

```c
        case 2:
            printf("USN    : ");   scanf("%s", usn);
            printf("Name   : ");   scanf("%s", name);
            printf("Progrm : ");   scanf("%s", programme);
            printf("Phone  : ");   scanf("%s", phone);
            printf("Sem    : ");   scanf("%d", &sem);

            first = insert_rear(usn, name, programme, phone, sem, first);
            break;


        case 3:

            first = delete_front(first);
            break;

        case 4:

            first = delete_rear(first);
            break;

        case 5:
            display(first);
            break;

         default:
            exit(0);


    }
  }
}
```

Output:

```
 1:Front  insert   2:rear  insert
3:Front  delete   4:rear  delete
5:  Display   6:exit
Enter  your  choice   :1
USN: 1va24cs001
Name: aaa
programme: cs
phone: 9876543210
sem: 1


 1:Front  insert   2:rear  insert
3:Front  delete   4:rear  delete
5:  Display   6:exit
Enter  your  choice   :1
USN: 1va24cs002
Name: bbbb
programme: cs
phone: 9873210654
sem: 2


 1:Front  insert   2:rear  insert
3:Front  delete   4:rear  delete
5:  Display   6:exit
Enter  your  choice   :2
USN       : 1va24cs003
Name      : ccc
Progrm    : cs
Phone     : 9870123456
Sem       : 1

 1:Front insert  2:rear insert
3:Front delete  4:rear delete
5: Display  6:exit
Enter your choice  :5
Student list
usn       name     programme      phone      sem
1va24cs002   bbbb            cs          9873210654   2
1va24cs001   aaa             cs          9876543210   1
1va24cs003   ccc             cs          9870123456   1


 1:Front insert  2:rear insert
3:Front delete  4:rear delete
5: Display  6:exit
Enter your choice  :3
 The student with usn 1va24cs002 is deleted


 1:Front insert  2:rear insert
3:Front delete  4:rear delete
5: Display  6:exit
Enter your choice  :4
The student with usn 1va24cs003 is deleted


 1:Front insert  2:rear insert
3:Front delete  4:rear delete
5: Display  6:exit
Enter your choice  :6

...Program finished with exit code 0
```

8. Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo
a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL e. Demonstrate how this DLL can be used as Double Ended Queue.
f. Exit

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

typedef struct
{
    char    ssn[10];
    char    name[10];
    char    dept[10];
    char    desg[10];
    char    ph[10];
    float   sal;
} EMPLOYEE;

struct node
{
    char    ssn[10];
    char    name[10];
    char    dept[10];
    char    desg[10];
    char    ph[10];
    float   sal;

    struct node *llink;
    struct node *rlink;
};

typedef struct node *NODE;

NODE getnode()
{
    NODE x;

    x = (NODE) malloc(sizeof(struct node));
```

```c
    if (x == NULL)
    {
        printf("Not enough memory\n");
        exit(0);
    }

    return x;
}


void display(NODE head)
{
    NODE cur;

    if (head->rlink == head)
    {
        printf("Empty\n");
        return;
    }

    printf("SSN   NAME  DEPT  DESGIN  PHONE  SAL\n");
    cur = head->rlink;

    while (cur != head)
    {
        printf("%-10s%-10s%-10s%-10s%-14s%f\n",cur->ssn, cur->name, cur->dept, cur->desg, cur->ph, cur->sal);
        cur = cur->rlink;
    }
}

int count_node(NODE head)
{
    NODE cur;
    int  count;

    if (head->rlink == head) return 0;

    cur = head->rlink;

    count = 0;

    while (cur != head)
```

```c
    {
        count++;
        cur = cur->rlink;
    }

    return count;
}

NODE insert_rear(EMPLOYEE emp, NODE head)
{
    NODE last, temp;

    temp = getnode();
    strcpy(temp->ssn, emp.ssn);
    strcpy(temp->name, emp.name);
    strcpy(temp->dept, emp.dept);
    strcpy(temp->desg, emp.desg);
    strcpy(temp->ph, emp.ph);
    temp->sal = emp.sal;

    // Get the address of the last node
    last = head->llink;

    // Insert the node at the end
    last->rlink = temp;
    temp->llink = last;
    temp->rlink = head;
    head->llink = temp;

    return head;
}

NODE insert_front(EMPLOYEE emp, NODE head)
{
    NODE temp, first;

    temp = getnode();
    strcpy(temp->ssn, emp.ssn);
    strcpy(temp->name, emp.name);
    strcpy(temp->dept, emp.dept);
    strcpy(temp->desg, emp.desg);
    strcpy(temp->ph, emp.ph);
    temp->sal = emp.sal;
```

```c
    // Get the address of the first node
    first = head->rlink;

    // Insert the node between head and first node
    temp->rlink = first;
    first->llink = temp;

    head->rlink = temp;
    temp->llink = head;

    return head;
}

NODE delete_rear(NODE head)
{
    NODE last, prev;

    // Check for empty list
    if (head->rlink == head)
    {
        printf("List is empty\n");
        return head;
    }

    // Obtain the address of the last node
    last = head->llink;

    // Obtain address of the last but one node
    prev = last->llink;

    // Make last but one node as the last node
    prev->rlink = head;
    head->llink = prev;

    printf("Item deleted = %s\n", last->ssn);
    free(last);

    return head;
}

NODE delete_front(NODE head)
{
```

```c
    NODE first, second;

    //Check for empty list */
    if (head->rlink == head)
    {
        printf("List is empty\n");
        return head;
    }

    first = head->rlink;         // Get the first node
    second = first->rlink;       // Get the second node

    //Make second node as the first node
    head->rlink = second;
    second->llink = head;

    printf("Item deleted = %s\n",first->ssn);
    free(first);

    return head;
}

void read_employee_details(EMPLOYEE *emp)
{
    printf("SSN        : ");         scanf("%s",emp->ssn);
    printf("Name       : ");         scanf("%s",emp->name);
    printf("Deptment   : ");         scanf("%s",emp->dept);
    printf("Designation : ");        scanf("%s",emp->desg);
    printf("Phone      : ");         scanf("%s",emp->ph);
    printf("Salary     : ");         scanf("%f",&emp->sal);
}

int main()
{
    int    choice, count;
    NODE   head;
    EMPLOYEE emp;

    head = getnode();
    head->rlink = head->llink = head;

    for (;;)
    {
```

```c
        printf("1:Insert Front  2:Insert Rear\n");
        printf("3:Delete Front  4:Delete Rear\n");
        printf("5:Display      6:Count  7: Exit : ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                read_employee_details(&emp);

                head = insert_front(emp, head);

                break;
            case 2:
                read_employee_details(&emp);

                head = insert_rear(emp, head);

                break;
            case 3:
                head = delete_front(head);
                break;

            case 4:
                head = delete_rear(head);
                break;
            case 5:
                display(head);
                break;

            case 6: count = count_node(head);
                printf("Number of nodes = %d\n", count);

                break;

            default:
                exit(0);
        }
    }
}
```

Output:

```
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 1
SSN          : 111
Name         : aaa
Deptment     : dept2
Designation  : tester
Phone        : 9876543210
Salary       : 20000
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 2
SSN          : 222
Name         : bbb
Deptment     : dept1
Designation  : developer
Phone        : 9032165478
Salary       : 30000
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 2
SSN          : 333
Name         : fff
Deptment     : dept4
Designation  : Manager
Phone        : 7894561230
Salary       : 50000
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 6
Number of nodes = 3
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 5
SSN     NAME     DEPT     DESGIN    PHONE    SAL
111      aaa        dept2      tester    9876543210     20000.000000
222      bbb        dept1    developer 9032165478     30000.000000
333      fff        dept4      Manager   7894561230     50000.000000
```

```
333        111       dept4      Manager     7894561230     50000.000
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 3
Item deleted = 111
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 3
Item deleted = 222
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 3
Item deleted = 333
1:Insert Front   2:Insert Rear
3:Delete Front   4:Delete Rear
5:Display        6:Count  7: Exit : 7


...Program finished with exit code 0
Press ENTER to exit console.
```

9. Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes.
a. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3
b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z) Support the program with appropriate functions for each of the above operations

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

struct node
{
    int     c;
    int     px;
    int     py;
    int     pz;
    struct node    *link;
};

typedef struct node * NODE;


float evaluate(float x, float y, float z, NODE head)
{
    float sum;
    NODE  cur;

    sum = 0;
    cur = head->link;
    while (cur != head)
    {
        sum = sum + cur->c * pow(x, cur->px) * pow(y,cur->py) * pow(z,    cur->pz);
        cur = cur->link;
    }

    return sum;
}

void print_polynomial(NODE head)
{
    NODE  p;
```

```c
    for (p = head->link; p != head; p = p->link)
    {
        if (p->c > 0)
            printf("+%dx^%dy^%dz^%d ", p->c, p->px, p->py, p->pz);
        else
            printf("%dx^%dy^%dz^%d ", p->c, p->px, p->py, p->pz);
    }

    printf("\n");
}

NODE getnode()
{
    NODE x;

    x = (NODE) malloc(sizeof(struct node));

    if (x == NULL)
    {
        printf("Not enough memory\n");
        exit(0);
    }

    return x;
}

NODE insert_rear(int c, int px, int py, int pz, NODE head)
{
    NODE cur, temp;

    temp = getnode();
    temp->c = c;
    temp->px = px;
    temp->py = py;
    temp->pz = pz;

    cur = head->link;
    while (cur->link != head)
    {
        cur = cur->link;
    }

    cur->link = temp;
```

```c
        temp->link = head;

        return head;
}

NODE read_polynomial()
{
        NODE head;
        int  c, px, py, pz;

        head = getnode();
        head->link = head;

        for(;;)
        {
            scanf("%d", &c);

            if (c == 0) break;

            scanf("%d", &px);
            scanf("%d", &py);
            scanf("%d", &pz);

            head = insert_rear(c, px, py, pz, head);
        }

        return head;
}

NODE add_2_polynomials(NODE h1, NODE h2)
{
        int  sum;
        NODE p, q, h3;

        h3 = getnode();
        h3->link = h3;

        for (p = h1->link; p != h1; p = p->link)
        {
            for (q = h2->link; q != h2; q = q->link)
            {
                if (p->px == q->px && p->py == q->py && p->pz == q->pz)
                {
```

```c
                    sum = p->c + q->c;
                    q->c = 0;

                    if (sum != 0) h3 = insert_rear(sum, p->px, p->py, p->pz, h3);

                    break;
                }
            }

            if (q == h2) h3 = insert_rear(p->c, p->px, p->py, p->pz, h3);
        }

        for (q = h2->link; q != h2; q = q->link)
        {
            if (q->c == 0) continue;

            h3 = insert_rear(q->c, q->px, q->py, q->pz,h3);
        }

        return h3;
}

int main()
{
        NODE  h1, h2, h3;
        int   choice;
        float x, y, z, sum;

        for(;;)
        {
            printf("1:Add  2:Evaluate  3:Exit : ");
            scanf("%d", &choice);

            switch(choice)
            {
                case 1:
                    printf("Enter first polynomial : ");
                    h1 = read_polynomial();

                    printf("Enter second polynomial : ");
                    h2 = read_polynomial();

                    printf("Poly1: ");
```

```
                print_polynomial(h1);

                printf("Poly2: ");
                print_polynomial(h2);

                h3 = add_2_polynomials(h1, h2);

                printf("Poly3: ");
                print_polynomial(h3);

                break;

            case 2:
                printf("Enter a polynomial");
                h1 = read_polynomial();

                printf("Enter x y and z : ");
                scanf("%f %f %f", &x, &y, &z);

                sum = evaluate(x, y, z, h1);

                printf("Result = %f\n", sum);

                break;

            default:
                exit(0);
        }
    }
}
```

Output:

```
/tmp/a.out
1:Add   2:Evaluate   3:Exit : 1
Enter first polynomial : 2 3 2 1 2 1 0 1 0
Enter second polynomial : 3 3 2 1 -2 1 0 1 0
Poly1: +2x^3y^2z^1 +2x^1y^0z^1
Poly2: +3x^3y^2z^1 -2x^1y^0z^1
Poly3: +5x^3y^2z^1
1:Add   2:Evaluate   3:Exit : 2
Enter a polynomial2 3 2 1 2 1 0 1 0
Enter x y and z : 2 3 4
Result = 592.000000
1:Add   2:Evaluate   3:Exit : |
```