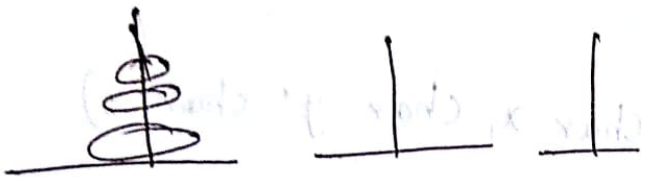
PART – A

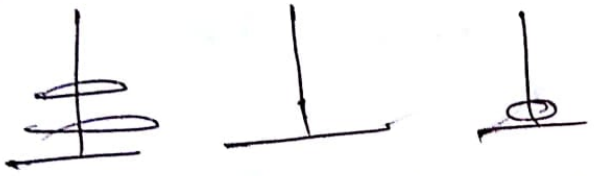
① write a c function to demostrate the movement of disks in ~~tasks~~ tower of vnai using recursue method. also show the pictorial representation of three disk movement of TOAs.

Sol:- The daigram below shows the move

①



②



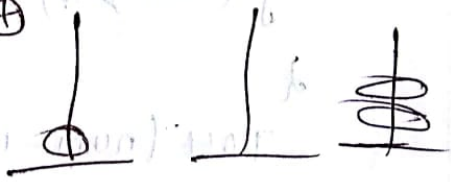
③

④

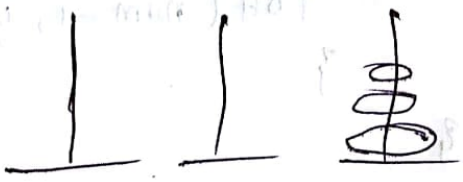
⑤

⑥

⑦



⑧



Programm :-

```
void TOH (int, char, char, char);
{
   int num;
   printf("\n Enter number of pdater: ");
   scanf(" %d", &num);
   TOH (num-1, 'A', 'B', 'B');
   return (0);
}
 void TOH (int num, char x, char y, char 3)
 {
    if (num>0)
    {
     TOH (num-1, x, z, y);
     printf(" more a disk from %d to %d x, z);
     printf(" \n %c -> %c", x, z);
      TOH (num-1, y, x, z);
    }
}
```

② write a recursive function to computer GCD of two numbers. also toce the function with GCD (84, 246)

Solu:- Recursive function:

$$GCD(m,n) = \begin{cases} GCD(n,m), & \text{if } n > m \\ m, & \text{if } n = 0 \\ GCD(n, m \bmod n), & \text{otherwise} \end{cases}$$

Trace :-

gcd (84, 246)
|
gcd (246, 84)
|
gcd (84, 246 mod 84)
|
gcd (84, 78)
|
gcd (78, 4)
|
return 6.

gcd (n, m)
gcd (n, mod (m, n))
gcd (n, m)
gcd (n, mod, (m, n))

⑧ By using parameter passing method, emplement a c program to demostrate the looking of a circuler Quie.

Solu:-

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
 int q full (count = qsize)
{
   return (count == qsize) ? 1:0;
}
 int q empty (int count)
{
   return (count ==0) q 1:0;
}
void q_ insert item, int qeuer[], int rear,
                                  int count)
{
   if (q full count)
   {
     printf (" overflow of queu \n");
     return;
   }
   rear = (rear+1) % q size;
   queu [rear] = item;
    count ++;
}
 void q_ delete (int queu[], int to front, int
                                  count)
{
   if (q empty (count))
   {
     printf (" Queu empty \n");
```

```
}
printf("The deleted element is %d", queu[front]);
        front = (front + 1) % q size)
        * count = 1;

}
void display (int queue[], int front, int count)
{
  int i, j;
  if(qempty (count))
  {
   printf("queu empty\n");
    return;
  }
  printf("content of queu is \n");
        i = front;
        for(j = 1 <= count; i++)
      {printf("%d", queu[i]);
       i=(i+1) % queu size
      }
      printf("\n")
}
void main()
{ int choice, item front, rear, count, queu[20];
  front = 0; rear = -1; count = 0;
  for(;;)
  {
    printf("1: insert 2: delete\n");
    printf("3: display 4: exit\n");
    printf("enter the choice\n");
    scanf("%d", & choice);
    switch (choice)
    {
```

```
        case 1: printf("enter the item to be insert\n");
                scanf("%d", &item);
                eq_insert(item, queu, &rear, count);
                break;

        case 2: eq_delete(queu, &front, &count);
                break;

        case 3: display(queu, front, count);
                break;

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

4. convert the infex expression ((A+B-C)*D) E+E) to postfix expression. write a c program to evaluate profex expression

Solu:- ((A+B-C)*D)'E+F)

((A+T1D)'E+F)

(A+T2)(E+F)

T3

(T3'E+F)

(TH+F)

T4F+

T3E'F+

AT2+E'F+

AT1D+TE'F+

ABC-D*+E'F+

T1 = BC

T2 = T1D

T3 = AT2+

T4 = T8E'

T4F+

⑤ Write the node structure for representing strings, linked. Also write c functions.

i) To insert the node from the front in the
ii) display the nodes in the SLL

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
    char USN[10], name[20], branch[10];
        int sem;
        long int ph;
        struct node link;
} node;
typedef struct node
{
   int count;
   NODE link;
} HEAD;
void int_front (HEAD *head);
void del_front (HEAD *head);
void display (HEAD *head);
void main()
{
   HEAD *head = (HEAD) malloc (SIZE of (HEAD));
   int choice;
   head -> count = 0;
   head -> link = NULL;
   for(;;)
   {
   printf(" Enter \n 1. Insert at front \n 2. delete at front \n
            3. display \n 4. Exit);
```

```
    scanf("%d", &choice);
    switch (choice)
    {
        case 1: ins_front (head); break;
        case 2: delete_front (head); break;
        case 3: display (head); break;
        case 4: Exit(0);
    }
  }
}
void ins_front (HEAD *head)
{
 NODE *newN = (NODE *) malloc (SIZE of (NODE));
 Printf ("Enter USN, name, branch, sem, phone of the stdent\n");
        newN -> link = head -> link;
        (head -> count)++;
         head -> link = newN;
}
 Void _delete front (HEAD * head)
{
   NODE *temp;
   if (head -> link = NULL)
   {
    printf ("list empty !!!\n");
    return;
   }
   temp = head -> link;
   printf ("deleted record :\n");
   printf ("%-s, \t %-s \t %d \t %d \t", (temp -> USN), (temp ->
          name), (temp -> branch), (temp -> sem), (temp -> ph));
      head -> link = temp -> link;
      (head -> count)--;
}    free (temp);
```

```
void display (HEAD * head)
{
    NODE temp;
    if (head -> link == NULL)
    {
      printf("list empty!!!\n");
      return;
    }
    else
    {
      printf("Number of nodes: %d\n", head -> count);
      printf("contents of the list\n");
      temp = head -> link;
      while (temp != NULL)
      {
      printf("%-5d\t%s\t %d\t %ld\t", (temp -> USN),
            (temp -> name), (temp -> branch), (temp -> sem),
                                    (temp -> ph));
            temp = temp -> link;
      }
    }
}
```

⑥ design, develop and implement a menu driven program in C for the following operation on doubly linked list of Employee data with the fields:
SSN, Name, Dept, destination, Sell, Phno

a) Create a DDL of N Employe data by using end insertion
b) display the struct of and count the numbers of nodes in it
c) perform insertion at end of dill
d) perform deletion at front of all

```
#include <stdio.h>
#include <stdlib.h>
typedef struct emp
{
    char SSN[10], name[20], department[10], designation[20];
    float salary;
    long int ph;
    struct emp *link, *link;
}NODE;
typedef struct head_node
{
  int count;
  NODE *link;
}HEAD;
Void ins_front (HEAD *head);
void ins_rear (HEAD *head);
void del_front (HEAD *head);
void del_rear (HEAD *head);
void display (HEAD *head);
Void main()
{
  HEAD *head = (HEAD *) malloc (sige of (HEAD));
```

```
int choice;
head -> count = 0;
head -> link = NULL;
for(;;)
{
  printf("enter\n1. insert at front\n2. insert at
        rear 3. delete at front\n4. delete at rear\n5.
        display the list. chose option 1 and 3 or 4 or 2
        and 3 or 4 for demostration of d queu\n7.
                                        EX\n4");
        scanf("%d", &choice);
        switch(choice)
        {
         case 1: ins-front(head);
                 break;
         case 2: ins-rear(head);
                 break;
         case 3: del-front(head);
         case 4: del-rear(head);
                 break;
         case 5: display(head);
         case 6: exit(0);
        }
    }
}
```

```
NODE *newN = (NODE)malloc(sizeof(NODE));
printf("Enter SSN, Employee name, department,
      destigration, salary, phone:\n4);
scanf("%s %s %s %f %d", (newN->ssn), (newN
-name), (newN -> department), (newN destignation
&newN -> salary) & (newN -> ph);
```

```
newN -> rlink = head -> link;
head -> link = newN;
head -> (count) ++;
return;
}
void ins_rear (HEAD *head)
{
  NODE *newN = (NODE *) malloc (size of (NODE));
  NODE temp
  printf(" Enter SSN, Employee, name, department, Salary
                                              phno");
        newN -> link = NULL;
        if (head -> link == NULL)
        {
          head -> link = newN;
          return;
        }
        temp = head -> link;
        while (temp -> link != NULL)
        {
          temp = temp -> link;
        }
        temp -> rlink = newN;
        (head -> count) ++;
}
void - del_front (HEAD *head)
{
  NODE *temp;
  if (head -> link == NULL)
  {
```

```
void display (HEAD * head)
{
  NODE *temp;
  if (head -> link == NULL)
  {
    printf("List empty!!!\n");
    return;
  }
  temp= head -> link;
  printf(" number of nodes:%d\n", (head -> count));
  printf(" Records:\n SSN\t name\t department
          designation salary phno\n");
    while (temp != NULL)
    {
     printf("%s\t %s\t %s\t %f %d %ld\n",(
          temp-SSN) (temp -> name), (temp -department
          (temp -> designation), (temp -> salary) (temp.
                                    ph));
          temp = temp -> rlink
        }
    }
```

PART - B

④. consider a senerio wher your web browser keep track of the web pages, browse suggest a method to implement a back button on the browser that takes you to the previous page. develop a data structure to store this information and enable the browser to display the previous page

Soln:-

```
# include <stdio.h>
# include <stdlib.h>
# define MAX 20
int push (int [max], int, int [], int *);
int pop (int [max], int, int [], int [], int *);
void main ()
{
    int stack[max], data, n size, sno;
    int top[10], both [10], limition [10];
    int i option, reply;
    printf(" C Language program \n");
    printf(" How many stack \n");
    scanf("%d", &n);
    SIZE = MAX \ n;
    both[0] = -1;
    for (i=1, i<n; i++)
        both [i] = both [-1] size;
    for(i=0; i<n-1; i++)
        limit [i] = both [i] + size;
    for (i=0; i<n ; i++)
```

```
top[i] = bott[p];
do
{
    printf("Language program to implement the
    printf("1. push\n");
    printf("2. pop\n");
    printf("3. exit\n");
    scanf("%d", option);
    {
        case 1 :
            printf("enter a logical stack
                number to %d)\n", n-1);
            scanf("%d", &sno);
            printf("enter a value.\n");
            scanf("%d", &data);
            reply = push(stack, sno, top,
                limit, &stack);
            if (reply == -1)
            printf("\n stack %d is
                full\n sno);
            else
            printf("%d is pushed in stack
                no %d\n", data, sno);
            break;
        case 2 : printf("enter a logical stack
                number co to %d);", n-1);
            scanf("%d", &sno);
            reply = pop(stack, sno, top, bott
                &data);
            if (reply == -1)
```

```
        printf("stack %d is empty \n", sno);
        else
        printf("%d is poped from stack no:%d\n", data, sno);
        break;
        case 3: break;
        }
    }
}
int push(int stack[max], int sno, int top[], int
                                    limit[], int *data)
{
    if (top[sno] == bott[sno])
        return(-1);
    else
    {
        *data = stack[top[sno]];
        top[sno]--;
        return;
    }
}
```

(8) case study : discuss the method to check if the SLL contains a paindrom were each node in the SLL store one charactors.

Solu:-

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
 int num;
 struct node *next;
};
 int create (struct node *);
int palin_check (struct node*, int);
void realease (struct node*);
int main()
{
   struct node *p = NULL;
   int result, count;
   printf(" enter data into the list \n");
   count = create (&p);
   result = palin_check (p, count);
   if (result == -1)
   {
   printf (The linked list a paindrom \n");
   }
   else
   {
   printf(" The linked list is not a palindrom \n");
   }
   release (&p);
   return (0);
}
```

```
int palin_check (struct node *p, int count)
{
   int i=0;
   struct node *front, *rear;
   while (i! =count/2)
   {
    front=rear=p;
    for (i=0; j<i+j ++)
   {
    front = front -> next;
   }
   for j=0; j< count-(i+1) ; j++)
   {
    rear= rear -> next;
   }
    if (front -> num! = rear -> num)
     {
      return 0;
     }
     else
     {
      i++;
     }
   }
   return;
}
int create [struct node * head]
{
   int ch, count=0;
   struct node *temp;
   do
   {
    printf (" Enter numbers ");
    scanf ("%d", &c);
    count ++;
    temp= struct node *) malloc (size of (struct node*));
    temp => num=c;
```

```
temp -> next = *head;

    *head = temp;
    printf("do you wish to continue I((o):");
    scanf("%d", &ch);
    } while (ch!=0);
    printf("\n");
    return count;
}

void release (struct node *head)
{
  struct node *temp = *head;
  while (*head) != NULL)
  {
    (*head) = (*head)->next
      free (temp);
      temp = *head;
    }
}
```