LAB 6 DSA

[A] Write a menu driven program to perform the following operations in a double linked
list by using suitable user defined functions for each case.

1. Create the list
2. Traverse the list in forward direction
3. Traverse the list in backward direction
4. Add a node after a given data item
5. Add a node before a given data item
6. Delete a node at a given position
7. Add a node a first node
8. Delete the first node
9. Reverse the content of the linked list by traversing each node only once.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int d;
   struct node * pre;
   struct node * ne;
}* start = '\0' , * last = '\0';
void create(int x)
{
   struct node * n = (struct node *)malloc(sizeof(struct node));
   n->d = x;
   n->ne = '\0';
   n->pre = '\0';
   if (start == '\0')
   {
      start = n;
      last = n;
   }
   else
   {
      last->ne = n;
      n->pre = last;
      last = n;
   }
}
void prfor()
{
    struct node * n = start;
    while (n != '\0')
    {
        printf("%i \n",n->d);
        n = n->ne;
    }
}
void prbac()
{
    struct node * n = last;
    while (n != '\0')
    {
        printf("%i \n",n->d);
        n = n->pre;
    }
}
void inaf(int x, int y)
{
    struct node * s = start, * n = (struct node *)malloc(sizeof(struct node));
    n->d = x;
    n->ne ='\0';
    n->pre = '\0';
    while (s->d != y && s != '\0'){s = s->ne;}
    if (s == last)
    {
        s->ne = n;
        n->pre = s;
        last = n;
    }
    else if (s != '\0')
    {
        n->ne = s->ne;
        n->pre = s;
        s->ne->pre = n;
        s->ne = n;
    }
    else{printf("Entered Element Not Present In List");}
}
void inbf(int a, int s)
{
           struct node * x = (struct node *)malloc(sizeof(struct node)),* n =
start;
           while(n->d != s && n != '\0'){n = n->ne;}
    if (n != '\0')
    {
        x->d = a;
        x->ne = '\0';
        x->pre = '\0';
        if (n->pre == '\0')
        {
           x->ne = n;
           n->pre = x;
           start = x;
        }
        else
        {
           x->ne = n;
           n->pre->ne = x;
           x->pre = n->pre;
           n->pre = x;
        }
    }
    else{printf("Entered Element Not Present In List");}
}
void delpos(int x)
{
           struct node * n = start;
           while(n->d != x && n != '\0'){n = n->ne;}
           if (n == last)
    {
        last = n->pre;
        n->pre->ne = '\0';
        free(n);
    }
    else if (n == start)
    {
        start = n->ne;
        n->ne->pre = '\0';
        free(n);
    }
```

```
    else if (n != '\0')
    {
            n->pre->ne = n->ne;
        n->ne->pre = n->pre;
            free(n);
    }
    else{printf("Entered Element Not Present In List");}
}
void addfirst(int a)
{
    struct node * n = (struct node *)malloc(sizeof(struct node));
    n->d = a;
    n->pre = '\0';
    n->ne = start;
    start->pre = n;
    start = n;
}
void delfirst()
{
    struct node * n = start;
    n->ne->pre = '\0';
    start = n->ne;
    free(n);
}
void rev()
{
    struct node * n = start, * s = last;
    while ((n != s) && (n->pre != s))
    {
        int t = n->d;
        n->d = s->d;
        s->d = t;
        n = n->ne;
        s = s->pre;
    }
}
void main()
{
    int n, s, z = 1, m;
    printf("Enter No. Of Nodes To Add :~ ");
```

```
scanf("%i",&n);
printf("Enter Values To Enter :~ ");
for (int i = 0 ; i < n ; i++)
{
   scanf("%i",&s);
   create(s);
}
printf("1-> Print List\n");
printf("2-> Print List In Rev Order\n");
printf("3-> Insert Node After Given Data\n");
printf("4-> Insert Node Before Given Data\n");
printf("5-> Delete Value\n");
printf("6-> Add First Node\n");
printf("7-> Delete First Node\n");
printf("8-> Reverse Nodes\n");
printf("9-> Exit Program\n");
while(z)
{
   printf("Enter Your Choice :~ ");
   scanf("%i",&n);
   if (n == 1){prfor();}
   else if (n == 2){prbac();}
   else if (n == 3)
   {
      printf("Enter Value :~ ");
      scanf("%i",&s);
      printf("Enter Data To Be Added After :~");
      scanf("%i",&m);
      inaf(s,m);
   }
   else if (n == 4)
   {
      printf("Enter Value :~ ");
      scanf("%i",&s);
      printf("Enter Data To Be Added Before :~");
      scanf("%i",&m);
      inbf(s,m);
   }
   else if (n == 5)
   {
```

```
            printf("Enter Value To Delete :~ ");
            scanf("%i",&s);
            delpos(s);
        }
        else if (n == 6)
        {
            printf("Enter Value To Add :~ ");
            scanf("%i",&s);
            addfirst(s);
        }
        else if (n == 7){delfirst();}
        else if (n == 8){rev();}
        else if (n == 9){z = 0;}
        else {printf("Enter Valid Command\n");}
    }
}
```

Output

```
Enter No. Of Nodes To Add :~ 3
Enter Values To Enter :~ 4
5
6
1-> Print List
2-> Print List In Rev Order
3-> Insert Node After Given Data
4-> Insert Node Before Given Data
5-> Delete Value
6-> Add First Node
7-> Delete First Node
8-> Reverse Nodes
9-> Exit Program
Enter Your Choice :~ 6
Enter Value To Add :~ 57
Enter Your Choice :~ 8
Enter Your Choice :~ 1
6
5
4
57
```

[B] Write a menu driven program to perform the following operations in a circular linked
list by using suitable user defined functions for each case.
1. Create the list
2. Traverse the list
3. Add a node a first node
4. Delete the first node

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int d;
        struct node * ne;
} * start = '\0', * end;
void create(int x)
{
        struct node * n = (struct node *)malloc(sizeof(struct node));
        n->d = x;
        n->ne = start;
        if(start == '\0')
        {
                start = n;
                end = n;
        }
        else
        {
                end->ne = n;
                end = n;
        }
}
void print()
{
        struct node * n = start;
        while(n->ne != start)
        {
                printf("%i\n",n->d);
                n = n->ne;
        }
   printf("%i\n",n->d);
}
```

```
void addf(int x)
{
   struct node * n = (struct node *)malloc(sizeof(struct node));
         n->d = x;
         n->ne = start;
   end->ne = n;
   start = n;
}
void delf()
{
   struct node * n = start;
   end->ne = n->ne;
   start = n->ne;
   free(n);
}
void main()
{
   int n, s, z = 1;
   printf("Enter No. Of Nodes To Add :~ ");
   scanf("%i",&n);
   printf("Enter Values To List :~ ");
   for (int i = 0 ; i < n ; i++)
   {
      scanf("%i",&s);
      create(s);
   }
   printf("1-> Print List\n");
   printf("2-> Add First Node\n");
   printf("3-> Delete First Node\n");
   printf("4-> Exit Program\n");
   while(z)
   {
      printf("Enter Your Choice :~ ");
      scanf("%i",&n);
      if (n == 1){print();}
      else if (n == 2)
      {
         printf("Enter Value To Add :~ ");
         scanf("%i",&s);
         addf(s);
```

```
        }
        else if (n == 3){delf();}
        else if (n == 4){z = 0;}
        else {printf("Enter Valid Command\n");}
    }
}
```

OUTPUT

```
Enter No. Of Nodes To Add :~ 3
Enter Values To List :~ 4
5
6
1-> Print List
2-> Add First Node
3-> Delete First Node
4-> Exit Program
Enter Your Choice :~ 1
4
5
6
```

[C] Write a menu driven program to perform the following operations in a header linked
list by using suitable user defined functions for each case. The head node keeps the
count of the total number of nodes in the list. The data node keeps the student
information: Name, Roll No, CGPA, Address_City, Branch.

1. Create
2. Display student information
3. Display the total number of nodes (in O(1) time)
4. Display the students’ details belonging to a particular branch
5. Display the students’ details securing &gt; 7.5 CGPA and belonging to a given
branch.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
   char name[16];
   int roll;
   float cgpa;
   char adr[16];
   int branch;
   struct node * ne;
} * end;
struct head
{
   int c;
   struct node * ne;
} * start;
void create()
{
   struct node * n = (struct node *)malloc(sizeof(struct node));
   printf("Enter Name :~ ");
   scanf("%s",&n->name);
   printf("Enter Roll No. :~ ");
   scanf("%i",&n->roll);
   printf("Enter C.G.P.A :~ ");
   scanf("%f",&n->cgpa);
   printf("Enter Address :~ ");
   scanf("%s",&n->adr);
   printf("Enter Branch No. :~ ");
   scanf("%i",&n->branch);
   n->ne = '\0';
   if (start->ne == '\0')
   {
      start->ne = n;
      end = n;
   }
   else
   {
      end->ne = n;
```

```
        end = n;
    }
}
void display()
{
    struct node * n = start->ne;
    while (n != '\0')
    {
        printf("Name :~ %s\n",n->name);
        printf("Roll No. :~ %i\n",n->roll);
        printf("C.G.P.A :~ %0.2f\n",n->cgpa);
        printf("Address :~ %s\n",n->adr);
        printf("Branch No. :~ %i\n\n",n->branch);
        n = n->ne;
    }
}
void disno(){printf("No. Of Records Present Are :~ %i\n",start->c);}
void disbac(int x)
{
    struct node * n = start->ne;
    while (n != '\0')
    {
        if (n->branch == x)
        {
            printf("Name :~ %s\n",n->name);
            printf("Roll No. :~ %i\n",n->roll);
            printf("C.G.P.A :~ %0.2f\n",n->cgpa);
            printf("Address :~ %s\n",n->adr);
            printf("Branch No. :~ %i\n",n->branch);
        }
        n = n->ne;
    }
}
void dis75(int x)
{
    struct node * n = start->ne;
    while (n != '\0')
    {
        if (n->branch == x && n->cgpa >= 7.5)
        {
            printf("Name :~ %s\n",n->name);
            printf("Roll No. :~ %i\n",n->roll);
            printf("C.G.P.A :~ %0.2f\n",n->cgpa);
            printf("Address :~ %s\n",n->adr);
            printf("Branch No. :~ %i\n",n->branch);
        }
        n = n->ne;
    }
}
void main()
{
    start = (struct head *)malloc(sizeof(struct head));
    start->c = 0;
    start->ne = '\0';
    int n, s, z = 1;
    printf("Enter No. Of Nodes To Add :~ ");
    scanf("%i",&n);
    for (int i = 0 ; i < n ; i++)
    {
        create();
        start->c++;
    }
    printf("1-> Print List\n");
    printf("2-> Print Count\n");
    printf("3-> Display Data Of Department\n");
    printf("4-> Display Data Of Department With 7.5+ C.G.P.A\n");
    printf("5-> Exit Program\n");
    while(z)
    {
        printf("Enter Your Choice :~ ");
        scanf("%i",&n);
        if (n == 1){display();}
        else if (n == 2){disno();}
        else if (n == 3)
        {
            printf("Enter Department No. :~ ");
            scanf("%i",&s);
            disbac(s);
        }
        else if (n == 4)
        {
           printf("Enter Department No. :~ ");
           scanf("%i",&s);
           dis75(s);
        }
        else if (n == 5){z = 0;}
        else {printf("Enter Valid Command\n");}
    }
}
```

OUTPUT

```
Enter No. Of Nodes To Add :~ 2
Enter Name :~ Rishikesh
Enter Roll No. :~ 2105734
Enter C.G.P.A :~ 9
Enter Address :~ bbsr
Enter Branch No. :~ 406
Enter Name :~ Avipsa
Enter Roll No. :~ 2105708
Enter C.G.P.A :~ 9.5
Enter Address :~ bbsr
Enter Branch No. :~ 406
1-> Print List
2-> Print Count
3-> Display Data Of Department
4-> Display Data Of Department With 7.5+ C.G.P.A
5-> Exit Program
Enter Your Choice :~ 3
Enter Department No. :~ 1
Enter Your Choice :~ 1
Name :~ Rishikesh
Roll No. :~ 2105734
C.G.P.A :~ 9.00
Address :~ bbsr
Branch No. :~ 406

Name :~ Avipsa
Roll No. :~ 2105708
C.G.P.A :~ 9.50
Address :~ bbsr
Branch No. :~ 406
```

[D] Write a program to represent a sparse matrix in three tuple format using linked list
and write addition function to perform addition.

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int val;
   int row;
   int column;
   struct Node *ne;
} * s = '\0';
void create(struct Node ** start, int nz,int rw, int c )
{
   struct Node *t, *r;
   t = *start;
   if (t == '\0')
   {
      t = (struct Node *) malloc (sizeof(struct Node));
      t->val = nz;
      t->row = rw;
      t->column = c;
      t->ne = '\0';
      *start = t;

   }
   else
   {
      while (t->ne != '\0')
         t = t->ne;
      r = (struct Node *) malloc (sizeof(struct Node));
      r->val = nz;
      r->row = rw;
      r->column = c;
      r->ne = '\0';
      t->ne = r;

   }
}
void PrintList(struct Node* start)
```

```
{
   struct Node *t, *r, *s;
   t = r = s = start;
   printf("Row_position :~   ");
   while(t != '\0')
   {
      printf("%d ", t->row);
      t = t->ne;
   }
   printf("\n");
   printf("Column_postion :~ ");
   while(r != '\0')
   {
      printf("%d ", r->column);
      r = r->ne;
   }
   printf("\n");
   printf("Value :~          ");
   while(s != '\0')
   {
      printf("%d ", s->val);
      s = s->ne;
   }
   printf("\n");
}
void add(struct Node * s1, struct Node * s2)
{
   struct Node * t1 = s1, * t2 = s2;
   while (t1 != '\0' && t2 != '\0')
   {
      if (t1->row < t2->row)
      {
         create(&s,t1->val,t1->row,t1->column);
         t1 = t1->ne;
      }
      else if (t1->row == t2->row)
      {
         if (t1->column < t2->column){create(&s,t1->val,t1->row,t1-
>column);}
```

```
         else if (t1->column == t2->column){create(&s,t1->val+t2->val,t1-
>row,t1->column);}
         else {create(&s,t2->val,t1->row,t2->column);}
         t1 = t1->ne;
         t2 = t2->ne;
      }
      else
      {
         create(&s,t2->val,t2->row,t2->column);
         t2 = t2->ne;
      }
   }
   while (t1 != '\0')
   {
      create(&s,t1->val,t1->row,t1->column);
      t1 = t1->ne;
   }
   while (t2 != '\0')
   {
      create(&s,t2->val,t2->row,t2->column);
      t2 = t2->ne;
   }
}
void main()
{
   int m1[4][5] =
   {
      {0 , 0 , 3 , 0 , 4 },
      {0 , 0 , 5 , 7 , 0 },
      {0 , 0 , 0 , 0 , 0 },
      {0 , 2 , 6 , 0 , 0 }
   };
   int m2[4][5] =
   {
      {5 , 0 , 3 , 0 , 8 },
      {0 , 0 , 5 , 0 , 0 },
      {0 , 9 , 0 , 0 , 0 },
      {0 , 0 , 6 , 0 , 3 }
   };
   struct Node * s1 = '\0', * s2 = '\0';
   for (int i = 0; i < 4; i++)
      for (int j = 0; j < 5; j++)
         if (m1[i][j] != 0)
            create(&s1, m1[i][j], i, j);
   printf("First Matrix :~ \n");
   PrintList(s1);
   for (int i = 0; i < 4; i++)
      for (int j = 0; j < 5; j++)
         if (m2[i][j] != 0)
            create(&s2, m2[i][j], i, j);
   printf("Second Matrix :~ \n");
   PrintList(s2);
   add(s1,s2);
   PrintList(s);
}
```

OUTPUT

```
First Matrix :~
Row_position :~   0 0 1 1 3 3
Column_postion :~ 2 4 2 3 1 2
Value :~          3 4 5 7 2 6
Second Matrix :~
Row_position :~   0 0 0 1 2 3 3
Column_postion :~ 0 2 4 2 1 2 4
Value :~          5 3 8 5 9 6 3
Row_position :~   0 0 0 1 1 2 3 3
Column_postion :~ 0 2 4 2 3 1 1 2
Value :~          5 3 8 10 7 9 2 6
```

[E] Write a program to store a polynomial in linked list and write multiplication function to
perform multiplication of two polynomials.

```
#include<stdio.h>
#include<stdlib.h>
struct poly
{
   int degree;
   int coeff;
};
void pro(int n1, int n2, struct poly poly1[10], struct poly poly2[10])
{
   int c = -1;
   struct poly product[100];
   for(int i = 0 ; i < n1 ; i++)
   {
       for (int j=0;j< n2 ;j++)
       {
           product[++c].degree=poly1[i].degree+poly2[j].degree;
           product[c].coeff=poly1[i].coeff*poly2[j].coeff;
       }
   }
   printf("\nThe Product Of Two Polynomials Is: \n");
   for(int i = 0 ; i <= c ; i++)
   {
       if(product[i].degree==0){printf("%d ",product[i].coeff);}
       else if(product[i].degree==1){printf("%dx ",product[i].coeff);}
       else{printf("%dx^%d ",product[i].coeff,product[i].degree);}
       if(i!=c){printf("+ ");}
   }
}
void main()
{
   struct poly poly1[10],poly2[10],product[100];
   int n1,n2,count=-1;
   int i,j;
   printf("\nEnter Number Of Terms Of 1st Polynomial: ");
   scanf("%d",&n1);
   for(i = 0 ; i < n1 ; i++)
   {
       printf("\nEnter Degree: ");
       scanf("%d",&poly1[i].degree);
       printf("\nEnter Coefficient: ");
       scanf("%d",&poly1[i].coeff);
   }
   printf("\nEnter Number Of Terms Of 2nd Polynomial: ");
   scanf("%d",&n2);
   for(i = 0 ; i < n2 ; i++)
   {
       printf("\nEnter Degree: ");
       scanf("%d",&poly2[i].degree);
       printf("\nEnter Coefficient: ");
       scanf("%d",&poly2[i].coeff);
   }
   pro(n1,n2,poly1,poly2);
}
```

OUTPUT

```
Enter Number Of Terms Of 1st Polynomial: 2

Enter Degree: 2

Enter Coefficient: 423

Enter Degree: 1

Enter Coefficient: 24

Enter Number Of Terms Of 2nd Polynomial: 2

Enter Degree: 2

Enter Coefficient: 567

Enter Degree: 1

Enter Coefficient: 23

The Product Of Two Polynomials Is:
239841x^4 + 9729x^3 + 13608x^3 + 552x^2
```

RISHIKESH
2105734
CSE 32