[07/12/24, 1:10:21 AM] Jyotheeswar: evelop 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*/

```
#include<stdio.h>
#include<stdlib.h>
struct DList
{
	int ssn;
	char name[20];
	char desg[20];
	char dept[20];
	int sal;
	char phno[11];
	struct DList *prev,*next;
};
typedef struct DList dnode;
dnode *start=NULL;
dnode *create()
{
	dnode *newnode;
	newnode=(dnode*)malloc(sizeof(dnode));
	if(newnode==NULL)
		printf("Memory Overflow\n");
	else
	{
		printf("\n Enter ssn, name, designation,department,salary, phone number");

scanf("%d%s%s%s%d%s",&newnode->ssn,newnode->name,newnode->desg,newnode->dept,
&newnode->sal,newnode->phno);
		newnode->prev=NULL;
		newnode->next=NULL;
	}
return newnode;
}
void insert_front()
{
	dnode *nn;
```

```
	nn=create();
	if(start==NULL)
		start=nn;
	else
	{
		nn->next=start;
		start->prev=nn;
		start=nn;
	}
}
void delete_front()
{
	dnode *temp=start;
	if(start==NULL)
		printf("\n List is empty");
	else if(start->next==NULL)//single node
	{
		start=NULL;
		free(temp);
	}
	else
	{
		start=start->next;
		start->prev=NULL;
		free(temp);
	}
}
void insert_end()
{
	dnode *nn,*temp;
	nn=create();
	if(start==NULL)
		start=nn;
	else
	{
		temp=start;
		while(temp->next!=NULL)
			temp=temp->next;
		temp->next=nn;
	nn->prev=temp;
	}
}
void delete_end()
{
```

```
	dnode *temp=start;
	if(start==NULL)
		printf("\n Empty list");
	else if(start->next == NULL)
	{
		start=NULL;
		free(temp);
	}
	else
	{
		while(temp->next != NULL)
			temp=temp->next;
		(temp->prev)->next=NULL;
		free(temp);
	}
}
void traverse()
{
	int c=0;
	dnode *temp=start;
	if(start==NULL)
		printf("\n Empty list");
	else
	{
		printf("\n The details are");
		while(temp!=NULL)
		{

printf("\n%d\t%s\t%s\t%s\t%d\t%s\t",temp->ssn,temp->name,temp->desg,temp->dept,temp->sal
,temp->phno);
			c++;
			temp=temp->next;
		}
		printf("\n Number of nodes is %d",c);
	}
}
int main()
{
	int n,m,i;
	while(1)
	{
		printf("\n Enter 1:create
list\n2:insert_front\n3:insert_end\n4:delete_front\n5:delete_end\n 6:Display");
		scanf("%d",&m);
```

```
                switch(m)
                {
                        case 1: printf("\n Enter n");
                                        scanf("%d",&n);
                                        for(i=0;i<n;i++)
                                        insert_end();
                                        break;
                        case 2: insert_front();break;
                        case 3: insert_end();break;
                        case 4: delete_front();break;
                        case 5: delete_end();break;
                        case 6: traverse();break;
                        default:exit(0);break;
                }
        }
        return 0;
}
```

/*. Develop a Program in C for the following operationson Singly Circular nexted List (SCLL) with header nodes:
a. Represent and Evaluate a Polynomial P(x,y,z) = 6x2 y2 z2 - 4yz5 + 3x3 yz+2xy5 z-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.*/

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct poly
{
 int cf,px,py,pz,used;
 struct poly *next;
};
typedef struct poly node;
node* create()
{
        node* newnode;
        newnode=(node*)malloc(sizeof(node));
        printf("Enter the polynomial term(coeff, power of x, y, z):");
        scanf("%d%d%d%d",&newnode->cf,&newnode->px,&newnode->py,&newnode->pz);
```

```
	newnode->used=0;
	newnode->next=NULL;
	return newnode;
}
void insert_front(node* head)
{
	node* nn;
	nn=create();
	nn->next = head->next;
	head->next = nn;
}
void read_poly(node *head)
{
	int n,i;
	printf("Enter the no. of terms in polynomial:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	insert_front(head);
}
void display(node *head)
{
	node *temp = head->next;
	while(temp!= head)
	{
		printf(" %+dx^%dy^%dz^%d",temp->cf,temp->px,temp->py,temp->pz);
		temp=temp->next;
	}
}
void add_poly(node* h1, node* h2, node* h3)
{
	node *t1=h1->next, *t2,*nn;
	while(t1!=h1)
	{
		nn=(node*)malloc(sizeof(node));
		*nn=*t1;
		t2=h2->next;
		while(t2!=h2)
		{
			if(t2->used==0 && t1->px==t2->px && t1->py==t2->py &&
t1->pz==t2->pz)
			{
				nn->cf+=t2->cf;
				t2->used=1;
			}
```

```
                t2=t2->next;
            }
            nn->next=h3->next;
            h3->next=nn;
            t1=t1->next;
        }
        t2=h2->next;
        while(t2!=h2)
        {
            if(t2->used==0)
            {
                nn=(node*)malloc(sizeof(node));
                *nn=*t2;
                nn->next=h3->next;
                h3->next=nn;
            }
            t2=t2->next;
        }
}
void evaluate(node *h)
{
        int x,y,z,sum=0;
        node *temp=h->next;
        printf("Enter the values of x, y, z:");
        scanf("%d%d%d",&x,&y,&z);
        while(temp!=h)
        {
            sum+=temp->cf*pow(x,temp->px)*pow(y,temp->py)*
            pow(z,temp->pz);
            temp=temp->next;
        }
        printf("Evaluated value is %d\n",sum);
}
int main()
{
        node *h1,*h2,*h3;
        h1 = (node*) malloc(sizeof(node));
        h1->next = h1;
        h2 = (node*) malloc(sizeof(node));
        h2->next = h2;
        h3 = (node*) malloc(sizeof(node));
        h3->next = h3;
        printf("First polynomial\n");
        read_poly(h1);
```

```
        printf("Second polynomial\n");
        read_poly(h2);
        add_poly(h1,h2,h3);
        printf("\nTHE FIRST POLY IS\n");
        display(h1);
        printf("\nTHE SEC POLY IS\n");
        display(h2);
        printf("\nADDition of TWO poly are\n");
        display(h3);
        evaluate(h3);
}
```

/*10. Develop a menu driven Program in C for the following operations on Binary Search Tree (BST)
of Integers.
a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the BST in Inorder, Preorder and Post Order
c. Search the BST for a given element (KEY) and report the appropriate message
d. Exit*/

```
#include<stdio.h>
#include<stdlib.h>
struct Tree
{
        int info;
        struct Tree *left,*right;
};
typedef struct Tree tnode;
tnode *root=NULL;
tnode* getnode()
{
        tnode *newnode;
        newnode=(tnode*)malloc(sizeof(tnode));
        printf("\n Enter the value");
        scanf("%d",&newnode->info);
        newnode->left=newnode->right=NULL;
        return(newnode);
}
void insert()
{
        tnode *nn,*temp=root,*prev;
        nn=getnode();
```

```
        if(root==NULL)
                root=nn;
        else
        {
                while(temp!=NULL)
                {
                        prev=temp;
                        if(nn->info < temp->info)
                                temp=temp->left;
                        else
                                temp=temp->right;
                }
                if(nn->info < prev->info)
                        prev->left=nn;
                else
                        prev->right=nn;
        }
}
void inorder(tnode *root)
{
        if(root!=NULL)
        {
                inorder(root->left);
                printf("%d\t",root->info);
                inorder(root->right);
        }
}
void preorder(tnode *root)
{
        if(root!=NULL)
        {
                printf("%d\t",root->info);
                preorder(root->left);
                preorder(root->right);
        }
}
void postorder(tnode *root)
{
        if(root!=NULL)
        {
                postorder(root->left);
                postorder(root->right);
                printf("%d\t",root->info);
        }
```

```
}
int search(int key)
{
	tnode *temp=root;
	while(temp!=NULL)
	{
		if(key==temp->info)
			return 1;
		else if(key<temp->info)
			temp=temp->left;
		else
			temp=temp->right;
	}
	return 0;
}
void display(tnode *root,int height)
{
	int i;
	if(root!=NULL)
	{
		display(root->right,height+1);
		for(i=0;i<height;i++)
			printf("\t");
		printf("%d\n",root->info);
		display(root->left,height+1);
	}
}
int main()
{
	int ch,key,n,i;
	while(1)
	{
		printf("\n Enter\n 1:insert\t 2:traverse\t 3:search\t 4:display\t 5:exit \t");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: printf("\n Enter no of terms");
			scanf("%d",&n);
			for(i=0;i<n;i++)
				insert();
			break;
			case 2: if(root==NULL)
					printf("\n Tree empty");
				else
```

```
                                        {
                                                printf("\nInorder traversal:\t");
                                                inorder(root);
                                                printf("\nPreorder traversal:\t");
                                                preorder(root);
                                                printf("\nPostorder traversal:\t");
                                                postorder(root);
                                        }
                                        break;
                        case 3: printf("\n Enter the key");
                                        scanf("%d",&key);
                                        if(search(key))
                                                printf("Search is successful");
                                        else
                                                printf("Search is unsuccessful");
                                        break;

                        case 4: display(root,0);
                                        break;
                        case 5: return 0;
                }
        }
}
```

/*11. Develop a Program in C for the following operations on Graph(G) of Cities:
a. Create a Graph of N cities using Adjacency Matrix.
b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method*/

```
#include<stdio.h>
int a[10][10],visit[10],n;
void DFS(int s) {
int i;
visit[s]=1;
for(i=1;i<=n;i++) {
if((a[s][i]==1)&&(visit[i]==0))
DFS(i);
}
}
int main() {
int i,s,j;
```

```
printf("No of vertices\n");
scanf("%d",&n);
printf("\n Enter adjacency matrix");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
}
printf("\n Enter the source node");
scanf("%d",&s);
DFS(s);
printf("\n Nodes reachable from %d are \n",s);
for(i=1;i<=n;i++)
if(visit[i]==1 && i!=s)
printf("%d\t",i);
return 0;
}
```

/*Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function H: K ?L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.*/

```
#include<stdio.h>
#include<stdlib.h>
int L[100],max=10;
//float A = 0.1352;
void display()
{
int i;
printf("\n Hash table contents are ");
printf("\n Index\tdata\n");
for(i=0;i<max;i++)
printf("\n%d\t%d\n",i,L[i]);
}
void linear_probe(int addr,int num)
{
```

```
int i;
if(L[addr]==-1)
L[addr]=num;
else
{
printf("\n Collision detected");
i=(addr+1)%max;
while(i!=addr)
{
if(L[i]==-1)
{
L[i]=num;
printf("\n Collision resolved through linear probe");
return;
}
else
i=(i+1)%max;
}
printf("\n Hash table is full");
display();
}
}
int main()
{
int i,num,addr,input;
for(i=0;i<max;i++)
L[i]=-1;
do
{
printf("\n Enter the number");
scanf("%d",&num);
addr=num%max;
//key = max * (num*A - (int)(num*A));
printf("%d, %d",num, addr);
linear_probe(addr,num);
display();
printf("\n Enter 1 to continue");
scanf("%d",&input);
}while(input==1);
return 0;
}
```

/*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*/

```
#include<stdio.h>
#include<stdlib.h>
struct SLL
{
char usn[11];
char name[30];
char branch[4];
int sem;
char phno[11];
struct SLL *next;
};
typedef struct SLL Node;
Node *start=NULL;
Node *create()
{
        Node *newNode;
        newNode=(Node*)malloc(sizeof(Node));
        if(newNode==NULL)
        printf("Memory overflow");
        else
        {
                printf("\n Enter USN, name, branch, sem, phone number\n");

scanf("%s%s%s%d%s",newNode->usn,newNode->name,newNode->branch,&newNode->sem,
newNode->phno);
                newNode->next=NULL;
                return newNode;
        }
}
void insert_front()
{
        Node *newnode;
        newnode=create();
        newnode->next=start;
        start=newnode;
}
void delete_front()
```

```
{
	Node *temp;
	if(start==NULL)
	printf("\n List is empty");
	else
	{
		temp=start;
		start=start->next;
		free(temp);
	}
}
void insert_end()
{
	Node *nn,*temp;
	nn=create();
	if(start==NULL)
		start=nn;
	else
	{
		temp=start;
		while(temp->next!=NULL)
			temp=temp->next;
		temp->next=nn;
	}
}
void delete_end()
{
	Node *temp=start,*prev;
	if(start==NULL)
	{
		printf("\n Empty list");
		return;
	}
	if(start->next==NULL)
		start=NULL;
	else
	{
		while(temp->next!=NULL)
		{
			prev=temp;
			temp=temp->next;
		}
		prev->next=NULL;
	}
```

```
        free(temp);
}
void display()
{
        int c=0;
        Node *temp;
        if(start==NULL)
                printf("\n Empty list");
        else
        {
                temp=start;
                printf("\n The details are");
                while(temp!=NULL)
                {

printf("\n%s\t%s\t%s\t%d\t%s\t",temp->usn,temp->name,temp->branch,temp->sem,temp->phno
);
                        c++;
                        temp=temp->next;
                }
                printf("\n Number of nodes is %d",c);
        }
}
int main()
{
        int n,m,i;
        while(1)
        {
                printf("\n Enter your choice \n1:insert_front\n 2:insert_end\n 3:delete_front\n
4:delete_end\n5:display");
                scanf("%d",&m);
                switch(m)
                {
                        case 1: printf("\n Enter n");
                                        scanf("%d",&n);
                                        for(i=0;i<n;i++)
                                                insert_front();//Inserting multiple nodes at once
                                        break;
                        case 2: insert_end();break;
                        case 3: delete_front();break;
                        case 4: delete_end();break;
                        case 5: display();break;
                        default:return 0;
                }
```

```
        }
}
```