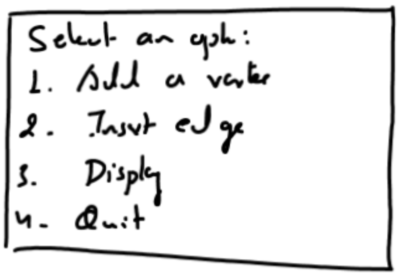
LINKED REPRESENTATION OF GRAPH



Select an optn:
1. Add a vertex
2. Insert edge
3. Display
4. Quit

```
struct edge
{
  char data;
  struct edge *next;
};
struct node
{
  char data;
  struct node *next;
  struct edge *enext;
};
void addvertex(struct node **,char);
void insertedge(struct node *,char,char);
void display(struct node*);
struct node * search(struct node *,char);
```

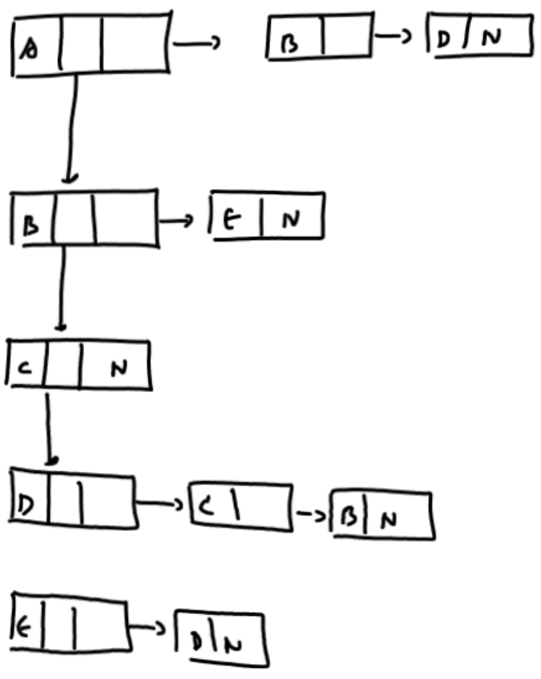
```
int main()
{
  struct node *start=NULL;
  char src,dest;
  int choice;
  do
  {
     printf("Select an operation:");
     printf("\n1.Add a vertex\n2.Imsert edge\n3.Display\n4.Quit");
     printf("\nEnter choice:");
     scanf("%d",&choice);
     switch(choice)
     {
      case 1:
               printf("Enter node data:");
               scanf(" %c",&src);
               addvertex(&start,src);
               break;
     case 2:
               printf("Enter src and dest vertices:");
               scanf(" %c %c",&src,&dest);
               insertedge(start,src,dest);
               break;
case 3:
                       display(start);
                       break;
case 4:
                      printf("Thank you for using the app!");
                      break;
default:
                      printf("Wrong choice! Try again");
 }
}while(choice!=4);
return 0;
}
```

```
void addvertex(struct node **ps,char ch)
{
    struct node *p,*temp;
    p=(struct node *)malloc(sizeof(struct node));
    if(p==NULL)
    {
       printf("Insufficient memory");
       return;
    }
   p->data=ch;
   p->next=NULL;
   p->enext=NULL;
   if(*ps==NULL)
   {
        *ps=p;
        return;
   }
   temp=*ps;
   while(temp->next!=NULL)
         temp=temp->next;
    temp->next=p;
}
```

```
struct node * search(struct node *p,char ch)
{
    while(p!=NULL)
    {
       if(p->data==ch)
            return p;
      p=p->next;
   }
return NULL;
}
```

```
void insertedge(struct node *p,char src,char dest)
{
   struct node *temp;
   struct edge *q,*r;
   if(p==NULL)
   {
      printf("Empty Graph");
      return;
   }
 temp=search(p,src);
  if(temp==NULL)
  {
     printf("Source vertex not found!");
     return;
   }
if(search(p,dest)==NULL)
 {
  printf("Dest vertex not found!");
  return;
}
q=(struct edge*)malloc(sizeof(struct edge));
q->data=dest;
q->next=NULL;
if(temp->enext==NULL)
{
  temp->enext=q;
  return;
}
r=temp->enext;
while(r->next!=NULL)
{
   r=r->next;
}
r->next=q;
}
```

Diagrammatic View Of display() function



A	:	B, D
B	:	E
C	:	
D	:	C, B
E	:	D

Assignments:
===========
1. WAF called delvertex() which should delete a vertex

2. WAF called deledge() which should delete abn edge from the graph.