deleting alternative

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
typedef struct node NODE;
struct llist
{
   NODE *head;
};
typedef struct llist LL;
void init(LL*);
void insert(LL*);
void rem(LL*);
void display(LL*);
int main()
{
   LL q;
   init(&q);
   insert(&q);
   printf("List before deleting\n");
   display(&q);
   rem(&q);
   printf("List after deleting\n");
   display(&q);

}
void init(LL *q)
```

```
{
   q->head=NULL;
}
void insert(LL *q)
{
   NODE *temp;
   int wish=1;
   do
   {
      temp=(NODE *)malloc(sizeof(NODE));
      printf("Enter data\n");
      scanf("%d",&(temp->data));
      temp->next=q->head;
      q->head=temp;
      printf("Do you want to enter more values\n");
      printf("Enter 1 for YES 0 for No\n");
      scanf("%d",&wish);
   }
   while(wish);
}
void rem(LL *q)
{
   int i=0;
   NODE *p=q->head;
   while(p!=NULL)
   {
      i++;
      p=p->next;
   }
   p=q->head;
   //printf("i value %d",i);
```

```
NODE *f=p;
NODE *prev=p->next;
p=p->next->next;
NODE *t=prev;
printf("Deleted %d\n",f->data);
free(f);
if(p==NULL || p->next==NULL)
{
   printf("Can't delete not sufficient data present\n");
   exit(1);
}
int count=0;
while(p!=NULL)
{
   if(count%2==0)
   {
      //printf("Hello");
      f=p;
      prev->next=p->next;
      prev=prev->next;
   }
   //printf("%d\n",p->data);
   p=p->next;
   if(count%2==0)
   {
      printf("Deleted %d\n",f->data);
      free(f);
   }
   if(i%2==0 && prev->next==NULL)
   {
      break;
```

```
        }
        if(i%2!=0 && count==i-2)
            prev->next=NULL;
        count++;
    }
    q->head=t;

}
void display(LL *q)
{
    NODE *p=q->head;
    if(p==NULL)
    {
        printf("Empty list\n");
    }
    while(p!=NULL)
    {
        printf("%d ",p->data);
        p=p->next;
    }
    printf("\n");
}
```

```
Microsoft Windows [Version 10.0.19042.1165]
Enter data
3
Do you want to enter more values
Enter 1 for YES 0 for No
0
List before deleting
3 7 4 5
Deleted 3
Deleted 4
List after deleting
7 5
```

swapping

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
typedef struct node NODE;
struct llist
{
   NODE *head;
};
typedef struct llist LL;
void init(LL*);
void insert(LL*);
void swap(LL*,int,int);
```

```
void display(LL*);
int main()
{
   LL q;
   init(&q);
   insert(&q);
   printf("List before swaping\n");
   display(&q);
   int a,b;
   printf("Enter the data values you want to be swapped\n");
   scanf("%d %d",&a,&b);
   swap(&q,a,b);
   printf("List after swapping\n");
   display(&q);

}
void init(LL *q)
{
   q->head=NULL;
}
void insert(LL *q)
{
   NODE *temp;
   int wish=1;
   do
   {
      temp=(NODE *)malloc(sizeof(NODE));
      printf("Enter data\n");
      scanf("%d",&(temp->data));
      temp->next=q->head;
      q->head=temp;
```

```
        printf("Do you want to enter more values\n");
        printf("Enter 1 for YES 0 for No\n");
        scanf("%d",&wish);
    }
    while(wish);
}
void swap(LL *q,int a,int b)
{
    NODE *prev1=NULL;
    NODE *prev2=NULL;
    NODE *prev=NULL;
    NODE *p=q->head;
    NODE *t;
    NODE *pres1;
    NODE *pres2;
    while(p!=NULL)
    {
        if(p->data==a)
            prev1=prev;
        if(p->data==b)
            prev2=prev;
        prev=p;
        p=p->next;
    }

    if(prev1==NULL && q->head->data==a)
    {
        prev1=q->head;
        pres1=prev1->next;
        pres2=prev2->next;
        t=pres2->next;
```

```
    q->head=pres2;
    pres2->next=pres1;
    prev2->next=prev1;
    prev1->next=t;
}
else if(prev2==NULL && q->head->data==b)
{
    prev2=q->head;
    pres2=prev2->next;
    pres1=prev1->next;
    t=pres1->next;
    q->head=pres1;
    pres1->next=pres2;
    prev1->next=prev2;
    prev2->next=t;
}
else if(prev2!=NULL && prev1!=NULL)
{
    pres1=prev1->next;
    pres2=prev2->next;
    t=pres2->next;
    prev1->next=pres2;
    pres2->next=pres1->next;
    prev2->next=pres1;
    pres1->next=t;
}
else
{
    printf("Entered value doesnt exist ");
    exit(0);
}
```

```
}
void display(LL *q)
{
   NODE *p=q->head;
   if(p==NULL)
   {
      printf("Empty list\n");
   }
   while(p!=NULL)
   {
      printf("%d ",p->data);
      p=p->next;
   }
   printf("\n");
}
```

```
Enter data
5
Do you want to enter more values
Enter 1 for YES 0 for No
0
List before swaping
10 9 71 5 5
Enter the data values you want to be swapped
10
71
List after swapping
71 9 10 5 5
```

polynomial

```
#include<stdio.h>
#include<stdlib.h>
struct poly
{
   int coeff;
   int px;
   int py;
   struct poly *next;
};
typedef struct poly POLY;
struct list
{
   POLY *head;
};
typedef struct list LL;
void init(LL*);
void accept(LL*);
POLY* res(LL*,LL*,LL*);
void display(LL*);
int main()
{
   LL q1,q2,q;
   init(&q1);
   init(&q2);
   init(&q);
   printf("Enter the first polynomial:\n");
   accept(&q1);
```

```
    printf("--------------------------\n");
    printf("Enter the second polynomial:\n");
    accept(&q2);
    q.head=res(&q1,&q2,&q);
    display(&q);
    return 0;
}
void init(LL *q)
{
    q->head=NULL;
}
void accept(LL *q)
{
    POLY *temp;
    char c;
    do
    {
        temp=(POLY *)malloc(sizeof(POLY));
        printf("Enter the coefficient:\n");
        scanf("%d",&(temp->coeff));
        printf("Enter the power of x:\n");
        scanf("%d",&(temp->px));
        printf("Enter the power of y:\n");
        scanf("%d",&(temp->py));
        temp->next=q->head;
        q->head=temp;
        fflush(stdin);
        printf("Do you want to enter more values\n");
        printf("Enter Y for yes N for No:\n");
        c=getchar();
        fflush(stdin);
```

```
    }
    while(c=='Y');
}
POLY* res(LL *q1,LL *q2,LL *q)
{
    POLY *p1=q1->head;
    POLY *p2=q2->head;
    POLY *p3=q->head;
    POLY *t=(POLY *)malloc(sizeof(POLY));
    while(p1!=NULL && p2!=NULL)
    {
        if((p1->px == p2->px) && (p1->py == p2->py))
        {
            t->coeff=(p1->coeff)+(p2->coeff);
            t->px=p1->px;
            t->py=p1->py;
            t->next=p3;
            p3=t;
            p1=p1->next;
            p2=p2->next;
        }
        else if((p1->px + p1->py)<=(p2->px + p2->py))
        {
            t->coeff=(p1->coeff);
            t->px=p1->px;
            t->py=p1->py;
            t->next=p3;
            p3=t;
            p1=p1->next;
        }
        else if((p1->px + p1->py)>(p2->px + p2->py))
```

```
    {
       t->coeff=(p2->coeff);
       t->px=p2->px;
       t->py=p2->py;
       t->next=p3;
       p3=t;
       p2=p2->next;
    }
    t=(POLY *)malloc(sizeof(POLY));
}
while(p1!=NULL)
{
    t->coeff=(p1->coeff);
    t->px=p1->px;
    t->py=p1->py;
    t->next=p3;
    p3=t;
    p1=p1->next;
    t=(POLY *)malloc(sizeof(POLY));
}
while(p2!=NULL)
{
    t->coeff=(p2->coeff);
    t->px=p2->px;
    t->py=p2->py;
    t->next=p3;
    p3=t;
    p2=p2->next;
    t=(POLY *)malloc(sizeof(POLY));
}
return p3;
```

```
}
void display(LL *q)
{
   POLY *p=q->head;
   while(p!=NULL)
   {
      if(p->next==NULL)
         printf("%dx(%d)y(%d)",p->coeff,p->px,p->py);
      else
         printf("%dx(%d)y(%d) +",p->coeff,p->px,p->py);
      p=p->next;
   }
}
```

```
Enter the first polynomial:
Enter the coefficient:
5
Enter the power of x:
2
Enter the power of y:
2
Do you want to enter more values
Enter Y for yes N for No:
Y
Enter the coefficient:
4
Enter the power of x:
2
Enter the power of y:
3
Do you want to enter more values
Enter Y for yes N for No:
Y
Enter the coefficient:
7
Enter the power of x:
1
Enter the power of y:
2
Do you want to enter more values
Enter Y for yes N for No:
```

```
Do you want to enter more values
Enter Y for yes N for No:
Y
Enter the coefficient:
7
Enter the power of x:
1
Enter the power of y:
2
Do you want to enter more values
Enter Y for yes N for No:
N
---------------------------
Enter the second polynomial:
Enter the coefficient:
7
Enter the power of x:
2
Enter the power of y:
2
Do you want to enter more values
Enter Y for yes N for No:
```