Set A

2) Write a program that adds two single variable polynomials. Each polynomial should be represented as a list with linked list implementation.

```
#include <stdio.h>
typedef struct pnode
{
float coef;
int exp;
struct pnode *next;
}p;
p *getnode();
void main()
{
p *p1,*p2,*p3;

p *getpoly(),*add(p*,p*);

void display(p*);
clrscr();
printf("\n enter first polynomial");
p1=getpoly();
printf("\n enter second polynomial");
p2=getpoly();
printf("\nthe first polynomial is");
display(p1);
printf("\nthe second polynomial is");
display(p2);
p3=add(p1,p2);
printf("\naddition of two polynomial is :\n");
display(p3);

}
p *getpoly()
{
p *temp,*New,*last;
int flag,exp;
char ans;
float coef;
```

```
temp=NULL;
flag=1;
printf("\nenter the polynomial in descending order of exponent");
do
{
printf("\nenter the coef & exponent of a term");
scanf("%f%d",&coef,&exp);
New=getnode();
if(New==NULL)
printf("\nmemory cannot be allocated");
New->coef=coef;
New->exp=exp;
if(flag==1)
{
temp=New;
last=temp;
flag=0;
}
else
{
last->next=New;
last=New;
}
printf("\ndou want to more terms");
ans=getch();
}
while(ans=='y');
return(temp);
}
p *getnode()
{
p *temp;
temp=(p*) malloc (sizeof(p));
temp->next=NULL;
return(temp);
}
void display(p*head)
{
p*temp;
temp=head;
```

```
if(temp==NULL)
printf("\npolynomial empty");
while(temp->next!=NULL)
{
printf("%0.1fx^%d+",temp->coef,temp->exp);
temp=temp->next;
}
printf("\n%0.1fx^%d",temp->coef,temp->exp);
getch();
}
p*add(p*first,p*second)
{
        p *p1,*p2,*temp,*dummy;
        char ch;
        float coef;
        p *append(int,float,p*);
p1=first;
p2=second;
temp=(p*)malloc(sizeof(p));
if(temp==NULL)
printf("\nmemory cannot be allocated");
dummy=temp;
while(p1!=NULL&&p2!=NULL)
{
if(p1->exp==p2->exp)
{
coef=p1->coef+p2->coef;
temp=append(p1->exp,coef,temp);
p1=p1->next;
p2=p2->next;
}
else
if(p1->expexp)
{
coef=p2->coef;
temp=append(p2->exp,coef,temp);
p2=p2->next;
}
else
if(p1->exp>p2->exp)
```

```
{
coef=p1->coef;
temp=append(p1->exp,coef,temp);
p1=p1->next;
}
}
while(p1!=NULL)
{
temp=append(p1->exp,p1->coef,temp);
p1=p1->next;
}
while(p2!=NULL)
{
temp=append(p2->exp,p2->coef,temp);
p2=p2->next;
}
temp->next=NULL;
temp=dummy->next;
free(dummy);
return(temp);
}
p*append(int Exp,float Coef,p*temp)
{
p*New,*dum;
New=(p*)malloc(sizeof(p));
if(New==NULL)
printf("\ncannot be allocated");
New->exp=Exp;
New->coef=Coef;
New->next=NULL;
dum=temp;
dum->next=New;
dum=New;
return(dum);
}
```

Set B

1) Write a program that sorts the elements of linked list using any of sorting technique.

```
#include<stdio.h>
#include<stdlib.h>

/* structure for a node */
struct Node
{
	int data;
	struct Node *next;
};

/* Function to insert a node at the beginning of a linked list */
void insertAtTheBegin(struct Node **start_ref, int data);

/* Function to bubble sort the given linked list */
void bubbleSort(struct Node *start);

/* Function to swap data of two nodes a and b*/
void swap(struct Node *a, struct Node *b);

/* Function to print nodes in a given linked list */
void printList(struct Node *start);

int main()
{
	int arr[] = {12, 56, 2, 11, 1, 90};
	int list_size, i;

	/* start with empty linked list */
	struct Node *start = NULL;

	/* Create linked list from the array arr[].
	Created linked list will be 1->11->2->56->12 */
	for (i = 0; i< 6; i++)
		insertAtTheBegin(&start, arr[i]);

	/* print list before sorting */
	printf("\n Linked list before sorting ");
	printList(start);

	/* sort the linked list */
```

```
	bubbleSort(start);

	/* print list after sorting */
	printf("\n Linked list after sorting ");
	printList(start);

	getchar();
	return 0;
}


/* Function to insert a node at the beginning of a linked list */
void insertAtTheBegin(struct Node **start_ref, int data)
{
	struct Node *ptr1 = (struct Node*)malloc(sizeof(struct Node));
	ptr1->data = data;
	ptr1->next = *start_ref;
	*start_ref = ptr1;
}

/* Function to print nodes in a given linked list */
void printList(struct Node *start)
{
	struct Node *temp = start;
	printf("\n");
	while (temp!=NULL)
	{
		printf("%d ", temp->data);
		temp = temp->next;
	}
}

/* Bubble sort the given linked list */
void bubbleSort(struct Node *start)
{
	int swapped, i;
	struct Node *ptr1;
	struct Node *lptr = NULL;

	/* Checking for empty list */
```

```
    if (start == NULL)
        return;

    do
    {
        swapped = 0;
        ptr1 = start;

        while (ptr1->next != lptr)
        {
            if (ptr1->data > ptr1->next->data)
            {
                swap(ptr1, ptr1->next);
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    }
    while (swapped);
}

/* function to swap data of two nodes a and b*/
void swap(struct Node *a, struct Node *b)
{
    int temp = a->data;
    a->data = b->data;
    b->data = temp;
}
```