Name: Manthan.M. Sonawane
Panel: E Roll No: 17
ADS

Assignment-1

- Problem statement: Implement polynomial operations using circular linked list: create, display, addition and evaluation.

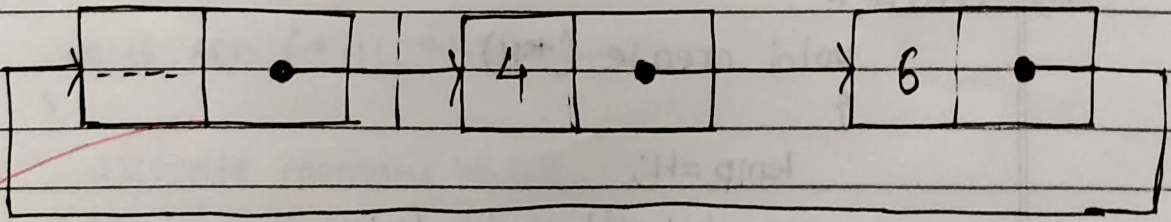
- Objectives:

a) To study the data structure - circular linked list
b) To study different operations that could be performed on CLL.
c) To study applications of CLL.

- Theory

1) Circular Linked List:

In a circular linked list, the last node be performed on CLL contains a pointer to the first node or the head node of the linked list. We traverse the CLL until we reach the same node where we started. It has no beginning or end. No NULL value is present in the next part of any of the node.



2) The difference betn SLL, CLL, DLL

- a) SLL only points to the node that is ahead of them and hence require only two minimum fields. The tail node of an SLL points to NULL as the tail node is the last node in the linked list.

- b) CLL also only points to the node that is ahead of them and hence only require two mininum fields but the tail node of the CLL points towards the head, hence making a complete circle.
- c) In DLL one node points to the next node as well as the node before it. Hence to create a DLL, a node must have three min. fields

3] Various operations on CLL

a) Insert
b) Delete
c) Reverse
d) Sort
e) Merge, etc.

- Platform

1) 64-bit open source Linux or its derivatives
2) Open source C++ programming tool like g++/Eclipse Editor.

- Pseudocode

A) Create :-

```
void create (*H)
{
    temp = H;
    repeat until choice 'y'
    {
    allocate memory to curr
    accept curr -> data
    curr -> next = H
    temp -> next = curr
```

```
        temp = curr
        read choice
      }
    }
```

B) Display :

```
      void Display (*H)
      {
      if H -> next == H
      printf("List is Empty")
      else
      {
        curr = Head -> next;
        while (curr != H)
        {
        print curr, curr -> data
        curr = curr -> next
        }
      }
    }
```

c) Add :

```
    void ADD (*H1, *H2)
    {
     allocate memory to H3
     Hedd3 -> exp = -1;
     t3 = H3;
     t1 = H1 -> next;
     t2 = H2 -> next;
    while (t2 -> exp! = -1 || t2 -> exp! = -1)
    {
```

```
if (t1 -> exp = t2 -> exp)
{
    allocate memory to temp
    Add t1 and t2 coeff in t3 coeff
    copy one of the exponent in t3 exp
    t3 -> next = temp;
    temp -> next = head3;
     t3 = temp;
   Move t1 to the next node
   Move t2 to the next node
 }
else
   if exp of p1 < exp of p2
   copy node of p2 to end of p3
else
   copy end of node p1 to p3
 }
}
```

- Time Complexity

i) create - O(n)

ii) Display - O(n)

iii) add - O(n) n1+n2

FAQ's

① Write an ADT for CLL

→ structure linked list (item)
declare create() → linked list
insert (item, linked list) → linked list
delete (linked list) → linked list
~~x add (linked list) → linked list → X~~
add (item, item) → linked list
ISEMPMS (linked list) → boolean;
For all L E linked list : i E item let
ISEMPS (CREATE) : : = true
ISEMPS (insert, (i, l)) : : = false
ISEMPS (DISPLAY) : : = true
end linked list

② How to perform multiplication of two polynomials!

→

(a) Multiply each term of one polynomial to all the other terms of other polynomial using the distributive law.
(b) Add the powers of the same variable using exponents
(c) simplify the newly obtained polynomial by adding or subtracting all the like terms.

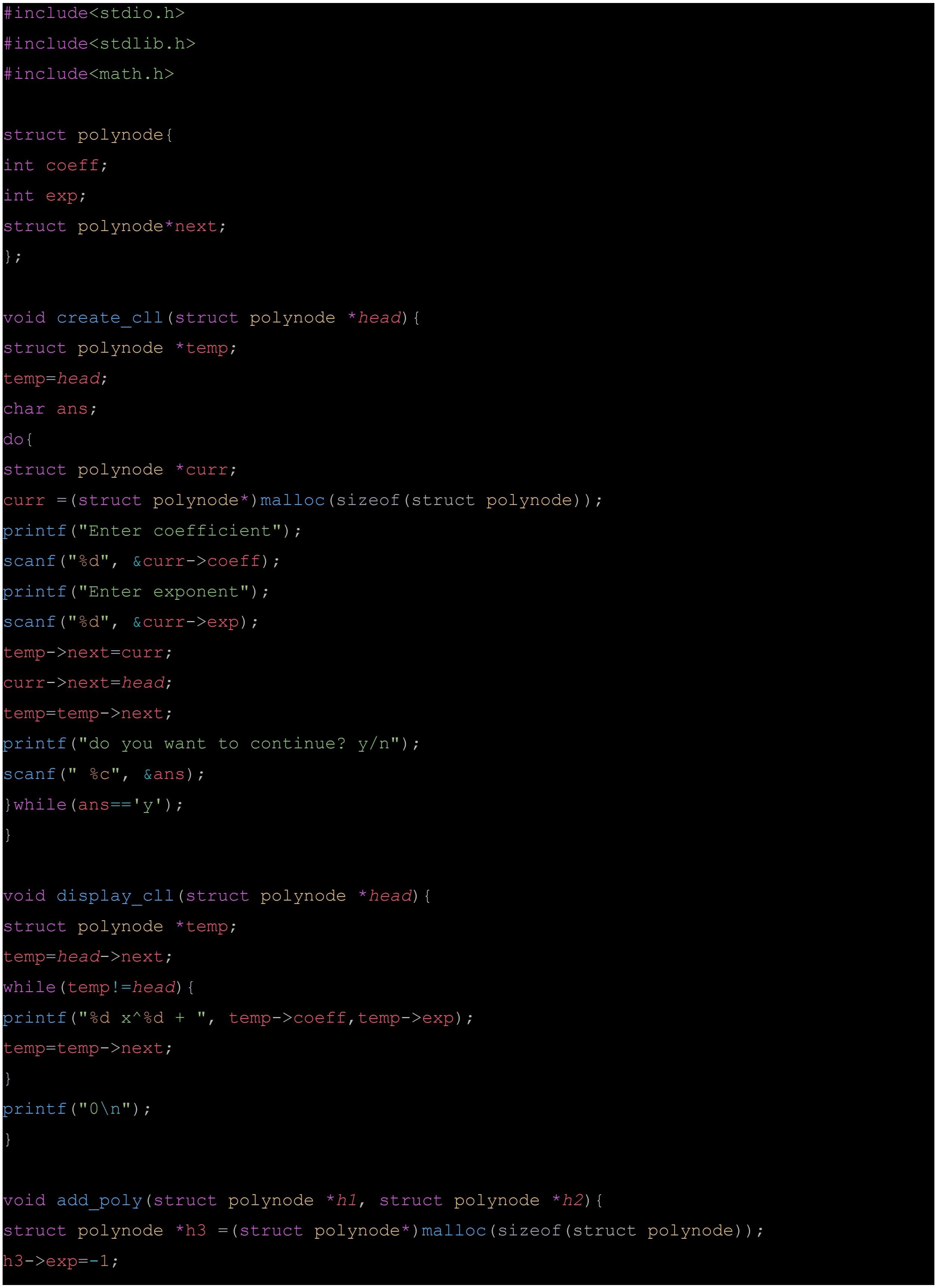
③ Write a polynomial addition algorithm if the terms are not sorted.

→ If the terms are not sorted in the polynomial, then we will just add a sort function to the code as follows:-

```
void sort (*H)
{
  len = len(H);
  prev = H;
  curr = H->next;
 for(i=0 ; i<len ; i++)
 {
  temp = curr->next
 if(curr -> exp > temp -> exp)
 {
   prev -> next-temp;
  curr -> next = temp ->next;
  temp -> next = curr;
  prev = temp;
 }
  else
 {
   prev = curr;
  curr = curr ->next;
  }
 }
}
```

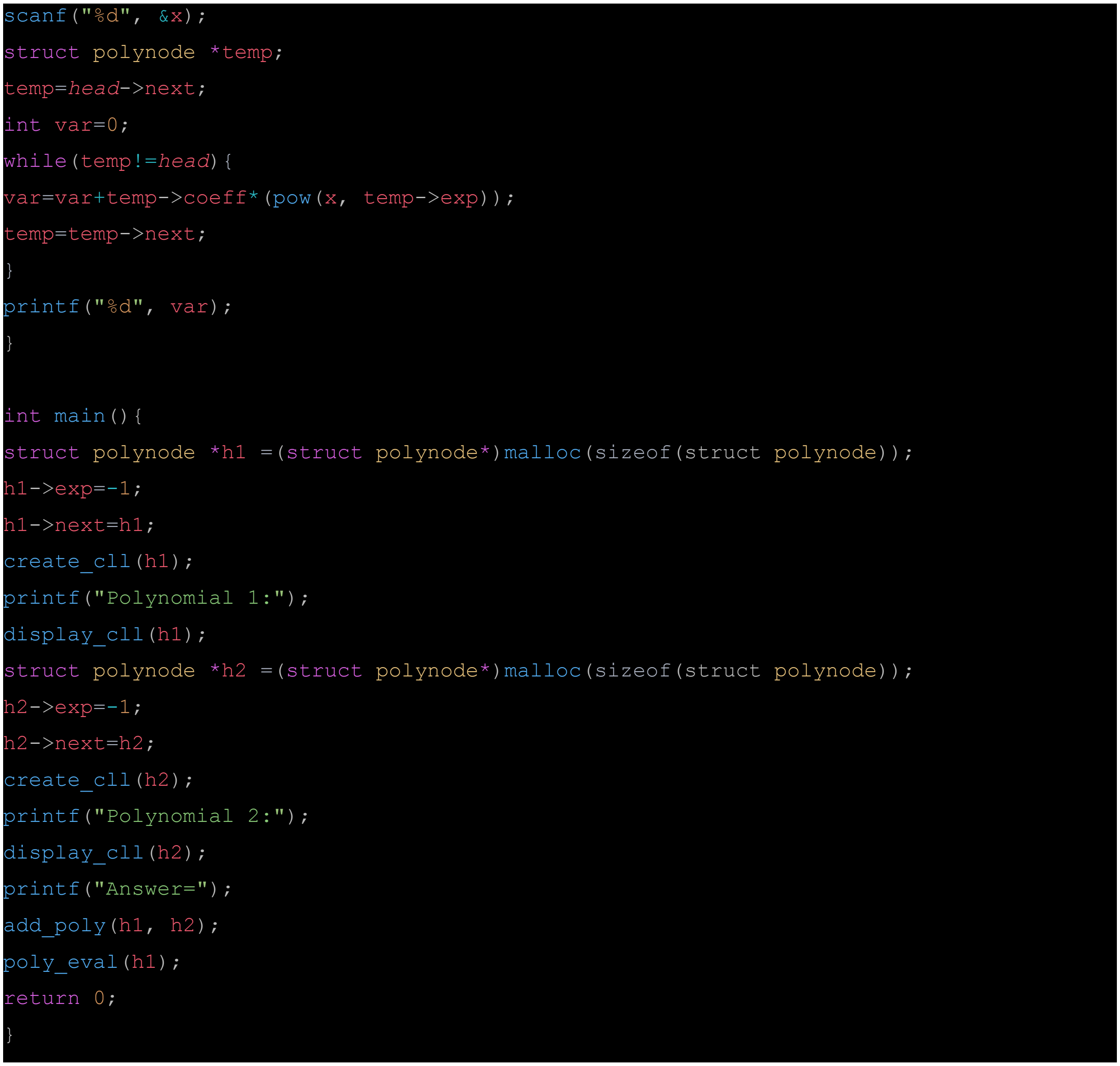
Time Complexity = $O(n^2)$

✓ 21/2/23



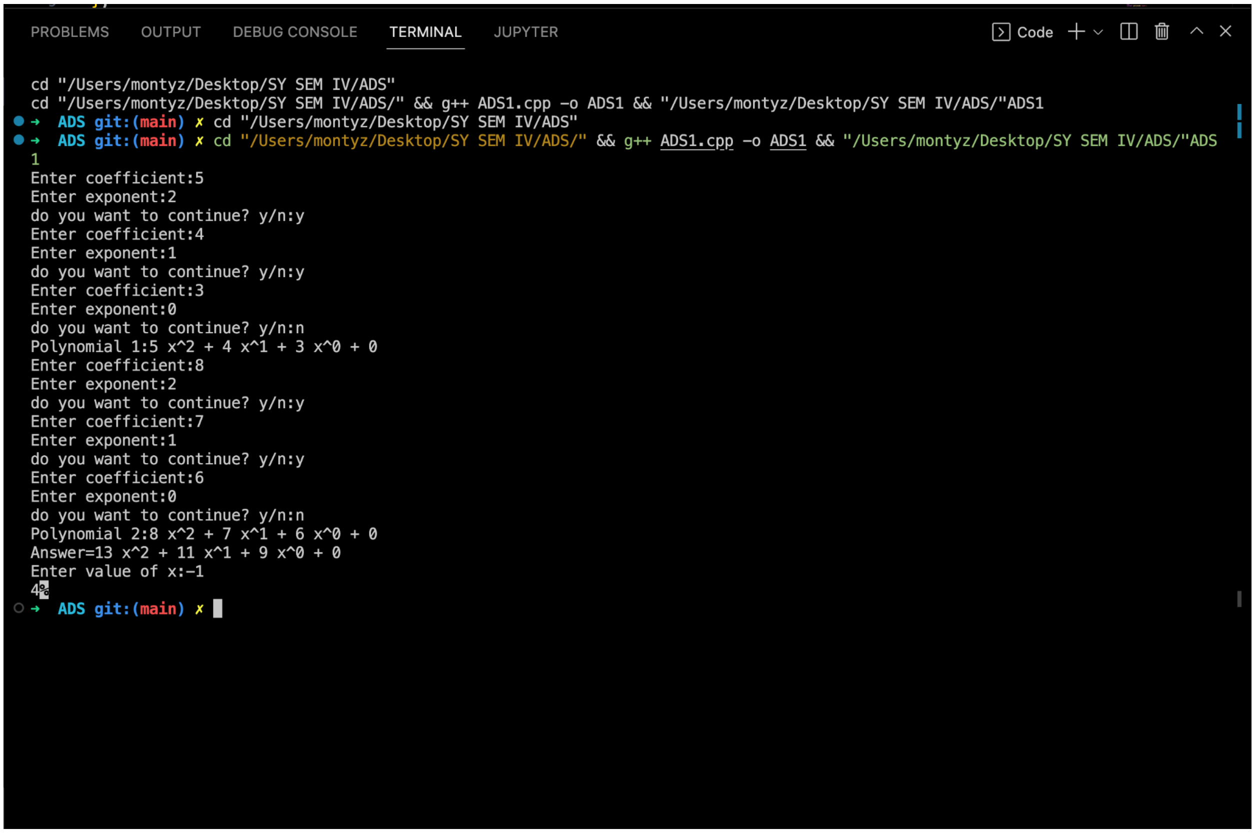
```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

struct polynode{
int coeff;
int exp;
struct polynode*next;
};

void create_cll(struct polynode *head){
struct polynode *temp;
temp=head;
char ans;
do{
struct polynode *curr;
curr =(struct polynode*)malloc(sizeof(struct polynode));
printf("Enter coefficient");
scanf("%d", &curr->coeff);
printf("Enter exponent");
scanf("%d", &curr->exp);
temp->next=curr;
curr->next=head;
temp=temp->next;
printf("do you want to continue? y/n");
scanf(" %c", &ans);
}while(ans=='y');
}

void display_cll(struct polynode *head){
struct polynode *temp;
temp=head->next;
while(temp!=head){
printf("%d x^%d + ", temp->coeff,temp->exp);
temp=temp->next;
}
printf("0\n");
}

void add_poly(struct polynode *h1, struct polynode *h2){
struct polynode *h3 =(struct polynode*)malloc(sizeof(struct polynode));
h3->exp=-1;
```

```
h3->next=h3;
struct polynode *t1, *t2, *t3;
t3=h3;
t1=h1->next;
t2=h2->next;
while(t1->exp!=-1||t2->exp!=-1){

if(t1->exp==t2->exp){
struct polynode *temp =(struct polynode*)malloc(sizeof(struct polynode));
temp->coeff=t1->coeff+t2->coeff;
temp->exp=t1->exp;
t3->next=temp;
temp->next=h3;
t3=temp;
t1=t1->next;
t2=t2->next;
}
else if((t1->exp)>(t2->exp)){
struct polynode *temp =(struct polynode*)malloc(sizeof(struct polynode));
temp->coeff=t1->coeff;
temp->exp=t1->exp;
t3->next=temp;
temp->next=h3;
t3=temp;
t1=t1->next;
}
else if((t1->exp)<(t2->exp)){
struct polynode *temp =(struct polynode*)malloc(sizeof(struct polynode));
temp->coeff=t2->coeff;
temp->exp=t2->exp;
t3->next=temp;
temp->next=h3;
t3=temp;
t2=t2->next;
}
}
display_cll(h3);
}

void poly_eval(struct polynode *head){
int x;
printf("Enter value of x:");
```



```
scanf("%d", &x);
struct polynode *temp;
temp=head->next;
int var=0;
while(temp!=head){
var=var+temp->coeff*(pow(x, temp->exp));
temp=temp->next;
}
printf("%d", var);
}

int main(){
struct polynode *h1 =(struct polynode*)malloc(sizeof(struct polynode));
h1->exp=-1;
h1->next=h1;
create_cll(h1);
printf("Polynomial 1:");
display_cll(h1);
struct polynode *h2 =(struct polynode*)malloc(sizeof(struct polynode));
h2->exp=-1;
h2->next=h2;
create_cll(h2);
printf("Polynomial 2:");
display_cll(h2);
printf("Answer=");
add_poly(h1, h2);
poly_eval(h1);
return 0;
}
```

Output



```
cd "/Users/montyz/Desktop/SY SEM IV/ADS"
cd "/Users/montyz/Desktop/SY SEM IV/ADS/" && g++ ADS1.cpp -o ADS1 && "/Users/montyz/Desktop/SY SEM IV/ADS/"ADS1
● → ADS git:(main) ✗ cd "/Users/montyz/Desktop/SY SEM IV/ADS"
● → ADS git:(main) ✗ cd "/Users/montyz/Desktop/SY SEM IV/ADS/" && g++ ADS1.cpp -o ADS1 && "/Users/montyz/Desktop/SY SEM IV/ADS/"ADS
1
Enter coefficient:5
Enter exponent:2
do you want to continue? y/n:y
Enter coefficient:4
Enter exponent:1
do you want to continue? y/n:y
Enter coefficient:3
Enter exponent:0
do you want to continue? y/n:n
Polynomial 1:5 x^2 + 4 x^1 + 3 x^0 + 0
Enter coefficient:8
Enter exponent:2
do you want to continue? y/n:y
Enter coefficient:7
Enter exponent:1
do you want to continue? y/n:y
Enter coefficient:6
Enter exponent:0
do you want to continue? y/n:n
Polynomial 2:8 x^2 + 7 x^1 + 6 x^0 + 0
Answer=13 x^2 + 11 x^1 + 9 x^0 + 0
Enter value of x:-1
4%
○ → ADS git:(main) ✗ █
```