**15B17CI371 – Data Structures Lab**

**ODD 2024**

**Week 1-LAB B**

**Practice Lab**

**9923103023-F1**

**[CO: C270.1]**

Instructions:

1. All students must save all their programs with the nomenclature

(Enroll No\_W1\_LabB\_QuestionNo.cpp). Also store the Outputs as well.

2. Upload them as per the instructions given by your lab faculty.

Concepts: Singly/ Doubly Linked List

Lab Questions:

1. Given a linked list, write functions to

a. Insert an element at the beginning of the linked list.

Ans :



#include<iostream>

using namespace std;

template <class T>

class LL;

template <class T>

class node

{

T data;

node<T> \*next=NULL;

int len=0;

node(T da)

{

data=da;

}

friend class LL<T>;

};

template <class T>

class LL

{

node<T>\* head=NULL;

public:

LL() {}

void inhead(T var)

{

node<T>\*temp=new node<T>(var);

if(head==NULL)

{

head =temp;

return;

}

temp->next=head;

head=temp;

}

void print()

{

node<T>\*temp = head;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp = temp->next;

}

}

void inspec(int index,T a)

{

node<T>\*temp = head;

for(int i=0; i<index-1; i++)

{

temp = temp->next;

}

node<T>\*add = new node<T>(a);

add->next =temp->next;

temp->next =add;

}

};

int main()

{

LL<int>\* ptr = new LL<int>;

int num,inp;

cout<<"enter number of elements :";

cin>>num;

for(int i=0;i<num;i++)

{

cin>>inp;

ptr->inhead(inp);

}

ptr->print();

//int n;

//cout<<"enter the index";

//cin>>n;

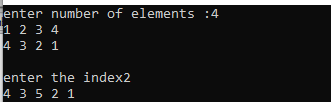
// ptr->inspec(n,5);

// ptr->print();

}

b. Insert an element at a specific location in a linked list.

Ans :



#include<iostream>

using namespace std;

template <class T>

class LL;

template <class T>

class node

{

T data;

node<T> \*next=NULL;

int len=0;

node(T da)

{

data=da;

}

friend class LL<T>;

};

template <class T>

class LL

{

node<T>\* head=NULL;

public:

LL() {}

void inhead(T var)

{

node<T>\*temp=new node<T>(var);

if(head==NULL)

{

head =temp;

return;

}

temp->next=head;

head=temp;

}

void print()

{

node<T>\*temp = head;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp = temp->next;

}

}

void inspec(int index,T a)

{

node<T>\*temp = head;

for(int i=0; i<index-1; i++)

{

temp = temp->next;

}

node<T>\*add = new node<T>(a);

add->next =temp->next;

temp->next =add;

}

};

int main()

{

LL<int>\* ptr = new LL<int>;

int num,inp;

cout<<"enter number of elements :";

cin>>num;

for(int i=0;i<num;i++)

{

cin>>inp;

ptr->inhead(inp);

}

ptr->print();

cout<<"\n\n";

int n;

cout<<"enter the index";

cin>>n;

ptr->inspec(n,5);

ptr->print();

}

c. Take input an integer number, split the number in its digits and stores the

digits in a linked list structure.

Ans :



#include<iostream>

using namespace std;

template <class T>

class LL;

template <class T>

class node

{

T data;

node<T> \*next=NULL;

int len=0;

node(T da)

{

data=da;

}

friend class LL<T>;

};

template <class T>

class LL

{

node<T>\* head=NULL;

public:

LL() {}

void inhead(T var)

{

node<T>\*temp=new node<T>(var);

if(head==NULL)

{

head =temp;

return;

}

temp->next=head;

head=temp;

}

void print()

{

node<T>\*temp = head;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp = temp->next;

}

}

void inspec(int index,T a)

{

node<T>\*temp = head;

for(int i=0; i<index-1; i++)

{

temp = temp->next;

}

node<T>\*add = new node<T>(a);

add->next =temp->next;

temp->next =add;

}

};

int main()

{

LL<int>\* ptr = new LL<int>;

int n,inp,digit;

cout<<"enter the number :";

cin>>n;

while(n!=0)

{

digit = n%10;

ptr->inhead(digit);

n/=10;

}

cout<<"\n";

ptr->print();

}

2. Write a program which reads a name and generates the link list of the characters

in that name. Later it removes the vowels from the link list and displays the

modified link list.

Ans : #include<iostream>

using namespace std;

#include<vector>

template<class T>

class LL;

template<class T>

class node

{

T data;

node<T> \* next=NULL;

public:

node(T data)

{

this->data=data;

}

friend class LL<T>;

};

template<class T>

class LL

{

node<T> \* head=NULL;

int len=0;

public:

void inath(T d)

{

node<T> \* temp=new node<T>(d);

if(!head)

{

head=temp;

len++;

return;

}

temp->next=head;

head=temp;

len++;

}

void print()

{

node<T> \* temp=head;

while(temp!=NULL)

{

cout<<temp->data<<' ';

temp=temp->next;

}

cout<<endl;

}

void inati(T d, int in)

{

if(in==0)

{

this->inath(d);

return;

}

node<T> \* temp=head;

node<T> \* n=new node<T>(d);

for(int i=0; i<in-1; i++)

{

temp=temp->next;

}

n->next=temp->next;

temp->next=n;

len++;

}

void inatt(T d)

{

if(!head)

{

inath(d);

return;

}

node<T> \* temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

node<T> \* n=new node<T>(d);

temp->next=n;

len++;

}

void remove\_v()

{

vector<node<T> \*> v;

node<T> \* temp=head;

while(temp!=NULL)

{

char a=temp->data;

if(a=='a' || a=='e' || a=='i' || a=='o' || a=='u')

{

v.push\_back(temp);

}

temp=temp->next;

}

temp=head;

for(int i=0; i<v.size(); i++)

{

this->dele(v[i]);

}

}

void dele(node<T> \* &del)

{

node<T> \* temp=head;

if(temp==del)

{

head=head->next;

delete del;

return;

}

while(temp!=NULL)

{

if(temp->next==del)

{

temp->next=temp->next->next;

delete del;

return;

}

temp=temp->next;

}

}

};

int main()

{

LL<char> \* l=new LL<char>();

string s;

cin>>s;

for(int i=0; i<s.length(); i++)

{

l->inatt(s[i]);

}

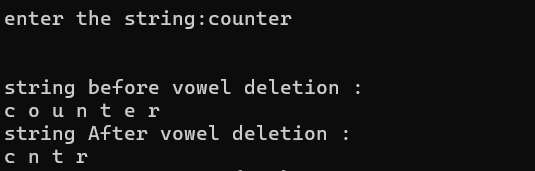
l->print();

l->remove\_v();

l->print();

return 0;

}



3. Create a link list of users supplied ten characters to store a name. Create a

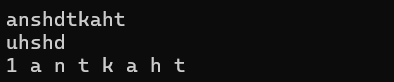
second link list of same type of user supplied five characters. Now using a

function remove(), traverse first link list and if any three consecutive characters

of second link list appears as consecutive characters of first link list, remove

those from first link list.

Ans :



#include<bits/stdc++.h>

using namespace std;

class node

{

public:

char data;

node\* next;

node(char val)

{

data = val;

next=nullptr;

}

};

node\* convert(string &arr,int n)

{

node\* head = new node(arr[0]);

node\* temp = head;

for(int i =1; i<n; i++)

{

node\* ptr = new node(arr[i]);

head->next = ptr;

head= ptr;

}

return temp;

}

void print(node\* head)

{

while(head!=nullptr)

{

cout<<head->data<<" ";

head=head->next;

}

cout<<endl;

}

void solve()

{

string s;

cin>>s;

int n=s.size();

string ss;

cin>>ss;

int m = ss.size();

if(n!=10 || m!=5)

{

cout<<"Invalid Input \n";

return ;

}

node\* big= convert(s,n);

node\* small=convert(ss,m);

node\* prev;

node\* dummy=new node('1');

dummy->next=big;

node\* temp1= dummy;

while(temp1!=nullptr)

{

char ch = temp1->data;

node\* temp2=small;

int i =0;

while(temp2!=nullptr && i<3)

{

if(temp1->data==temp2->data)

{

if((temp1->next->data==temp2->next->data)&&(temp2->next->next->data==temp1->next->next->data))

{

prev->next=prev->next->next->next->next;

}

}

else

{

temp2=temp2->next;

}

i++;

}

prev=temp1;

temp1=temp1->next;

}

print(dummy);

}

int main()

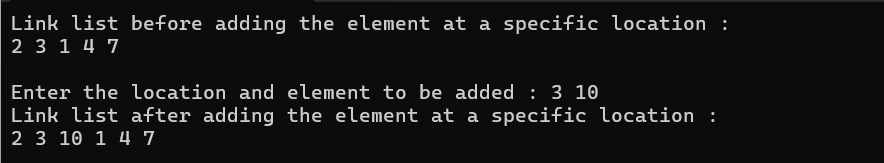
{

solve();

}

4. Write a program to insert an element at specific location in doubly linked list.

Ans :



#include<iostream>

using namespace std;

template<class T>

class LL;

template <class T>

class node

{

public:

T data;

node<T>\* previous;

node<T>\* next;

node(T data)

{

this->data=data;

this->next=nullptr;

this->previous=nullptr;

}

node() {}

friend class LL<T>;

};

template<class T>

class LL

{

public:

node<T>\* head = nullptr;

LL() {}

void inathead(T general)

{

node<T>\* member = new node<T>(general);

if(head==nullptr)

{

head=member;

return;

}

member->next = head;

head->previous=member;

head=member;

}

void printlist()

{

cout<<endl;

node<T>\* temp = head;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp=temp->next;

}

cout<<endl;

}

void inattail(T dat)

{

node<T>\* addnode = new node<T>(dat);

node<T>\* tempp = head;

while(tempp->next!=nullptr)

{

tempp = tempp->next;

}

tempp->next = addnode;

}

void inspeci(int index,T var)

{

int counter=1;

node<T>\* temp1 = head;

node<T>\* temp2 = head;

node<T>\* tail = head;

if(index == 0)

{

this->inathead(var);

return;

}

while(tail->next!=nullptr)

{

tail= tail->next;

counter++;

}

if(counter==index)

{

this->inattail(var);

return;

}

for(int i=0; i<index-1; i++)

{

temp1=temp1->next;

}

for(int i=0; i<index; i++)

{

temp2=temp2->next;

}

node<T>\* node1= new node<T>(var);

temp1->next=node1;

node1->previous=temp1;

node1->next=temp2;

temp2->previous=node1;

}

};

int main()

{

LL<int>\* list1 = new LL<int>;

node<int>\* head = nullptr;

list1->inathead(7);

list1->inathead(4);

list1->inathead(1);

list1->inathead(3);

list1->inathead(2);

list1->inattail(10);

cout<<"Link list before adding the element at a specific location : ";

list1->printlist();

int location;

int dat;

cout<<"\nEnter the location and element to be added : ";

cin>>location>>dat;

list1->inspeci(location-1,dat);

cout<<"Link list after adding the element at a specific location : ";

list1->printlist();

return 0;

}

5. Write a program to delete last element from the doubly linked list.

Ans :



#include<iostream>

using namespace std;

template<class T>

class LL;

template <class T>

class node

{

public:

T data;

node<T>\* previous;

node<T>\* next;

node(T data)

{

this->data=data;

this->next=nullptr;

this->previous=nullptr;

}

node() {}

friend class LL<T>;

~node()

{

int value = this->data;

if(next!=nullptr)

{

delete next;

next=NULL;

}

cout<<"Memory free for node with data : "<<value;

}

};

template<class T>

class LL

{

public:

node<T>\* head = nullptr;

LL() {}

void inathead(T general)

{

node<T>\* member = new node<T>(general);

if(head==nullptr)

{

head=member;

return;

}

member->next = head;

head->previous=member;

head=member;

}

void printlist()

{

cout<<endl;

node<T>\* temp = head;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp=temp->next;

}

cout<<endl;

}

void inattail(T dat)

{

node<T>\* addnode = new node<T>(dat);

node<T>\* tempp = head;

while(tempp->next!=nullptr)

{

tempp = tempp->next;

}

tempp->next = addnode;

}

void inspeci(int index,T var)

{

int counter=1;

node<T>\* temp1 = head;

node<T>\* temp2 = head;

node<T>\* tail = head;

if(index == 0)

{

this->inathead(var);

return;

}

while(tail->next!=nullptr)

{

tail= tail->next;

counter++;

}

if(counter==index)

{

this->inattail(var);

return;

}

for(int i=0; i<index-1; i++)

{

temp1=temp1->next;

}

for(int i=0; i<index; i++)

{

temp2=temp2->next;

}

node<T>\* node1= new node<T>(var);

temp1->next=node1;

node1->previous=temp1;

node1->next=temp2;

temp2->previous=node1;

}

void del\_Last(int position)

{

node<T>\* tem = head;

while(tem->next->next!=nullptr)

{

tem= tem->next;

}

node<T>\* cur = tem->next;

cur->previous = NULL;

tem->next=cur->next;

cur->next=NULL;

delete cur;

}

};

int main()

{

LL<int>\* list1 = new LL<int>;

node<int>\* head = nullptr;

int location;

int dat;

list1->inathead(7);

list1->inathead(4);

list1->inathead(1);

list1->inathead(3);

list1->inathead(2);

list1->inattail(10);

list1->printlist();

list1->del\_Last();

list1->printlist();

return 0;

}

6. Given a doubly linked list of any number of nodes, write a function

Extreme Swap(), which will swap values of the node at extreme pairs. For e.g., if

the node values of a doubly linked list are:

1 2 3 4 5 6 7 8

After first call, values will be

8 2 3 4 5 6 7 1

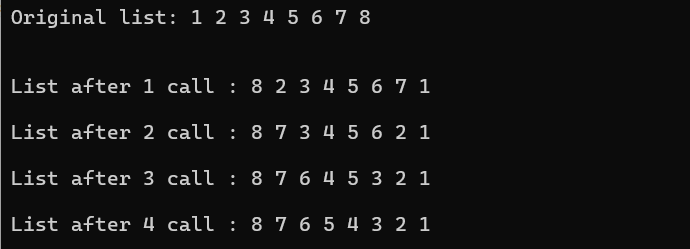
After second call, values will be

8 7 3 4 5 6 2 1

And finally, function will stop after fourth call, and the values will be

8 7 6 5 4 3 2 1

ANS :



#include <iostream>

struct Node

{

int data;

Node\* next;

Node\* prev;

Node(int val) : data(val), next(nullptr), prev(nullptr) {}

};

void swapNodes(Node\*& head,Node\*&tail, Node\* node1, Node\* node2)

{

if (node1 == node2 || node1 == nullptr || node2 == nullptr)

{

return;

}

if (node1->prev != nullptr)

{

node1->prev->next = node2;

}

else

{

head = node2;

tail =node1;

}

if (node2->prev != nullptr)

{

node2->prev->next = node1;

}

else

{

head = node1;

tail = node2;

}

if (node1->next != nullptr)

{

node1->next->prev = node2;

}

if (node2->next != nullptr)

{

node2->next->prev = node1;

}

std::swap(node1->next, node2->next);

std::swap(node1->prev, node2->prev);

}

void push(Node\*& head, int new\_data)

{

Node\* new\_node = new Node(new\_data);

Node\* last = head;

if (head == nullptr)

{

head = new\_node;

return;

}

while (last->next != nullptr)

{

last = last->next;

}

last->next = new\_node;

new\_node->prev = last;

}

void printList(Node\* node)

{

while (node != nullptr)

{

std::cout << node->data << " ";

node = node->next;

}

std::cout << std::endl;

}

int main()

{

Node\* head = nullptr;

int values[] = {1, 2, 3, 4, 5, 6, 7, 8};

for (int value : values)

{

push(head, value);

}

std::cout << "Original list: ";

printList(head);

Node\*tail =head;

while(tail->next!=nullptr)

{

tail=tail->next;

}

Node\* node1 = head;

Node\* node2 = tail;

std::cout<<"\n";

for(int i =0; i<4; i++)

{

swapNodes(head,tail, node1, node2);

std::cout << "\nList after " <<i+1<<" call : ";

printList(head);

node2= node2->next;

node1= node1->prev;

if(i>0)

{

node1=node1->next->next;

node2=node2->prev->prev;

}

if(i==2)

{

node1=node1->prev->prev->prev;

node2=node2->next->next->next;

}

}

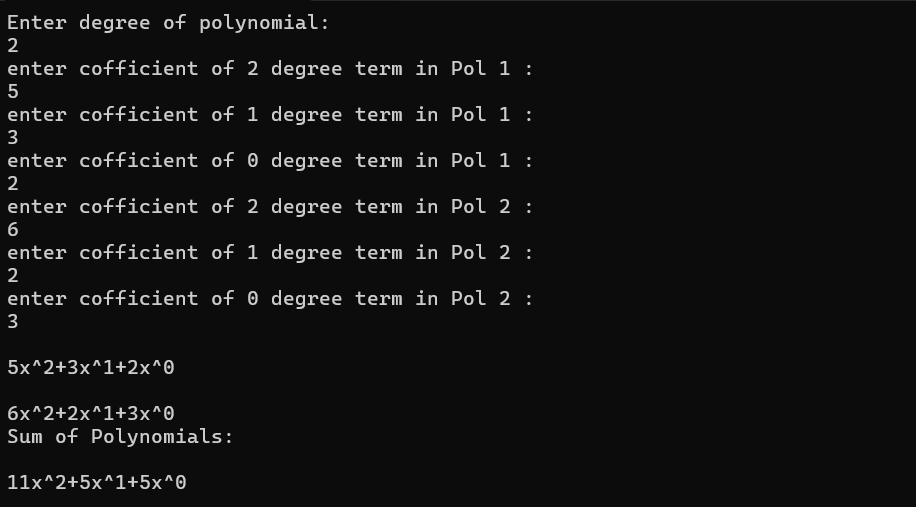
return 0;

}

7. Write a program to implement addition of two polynomials. Each node must

contain the value of the coefficient as well as its power as data components.

Ans :



#include<iostream>

#include<cstring>

using namespace std;

struct node

{

int data;

int degree;

struct node\* next;

};

void print(struct node\*head)

{

struct node\*ptr=head;

cout<<endl;

while(ptr->next!=NULL)

{

// cout<<"Degree: "<<ptr->degree<<" Coefficient: "<<ptr->data<<endl;

// ptr=ptr->next;

cout<<ptr->data<<"x^"<<ptr->degree<<"+";

ptr=ptr->next;

}

cout<<ptr->data<<"x^"<<ptr->degree;

cout<<endl;

}

struct node\* insertatend(struct node\*head, int data, int degree)

{

struct node\*p=new struct node;

struct node\*ptr=head;

p->data=data;

p->degree=degree;

if(ptr==NULL)

{

p->next=NULL;

head=p;

return head;

}

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=p;

p->next=NULL;

return head;

}

struct node\* addpol(struct node\*h1, struct node\*h2)

{

int degree,data;

struct node\*sum=new struct node;

sum=NULL;

struct node\*p=h1;

struct node\*q=h2;

while(p!=NULL)

{

degree=p->degree;

data=p->data + q->data;

sum=insertatend(sum,data,degree) ;

p=p->next;

q=q->next;

}

return sum;

}

int main()

{

struct node\*h1=NULL;

struct node\*h2= NULL;

int a,k,s;

cout<<"Enter degree of polynomial:\n";

cin>>s;

for(int i=s; i>=0; i--)

{

cout<<"enter cofficient of "<<i<<" degree term in Pol 1 :\n";

cin>>a;

h1=insertatend(h1,a,i);

}

for(int i=s; i>=0; i--)

{

cout<<"enter cofficient of "<<i<<" degree term in Pol 2 :\n";

cin>>a;

h2=insertatend(h2,a,i);

}

print(h1);

print (h2);

cout<<"Sum of Polynomials:\n";

struct node\*sum=addpol(h1,h2);

print(sum);

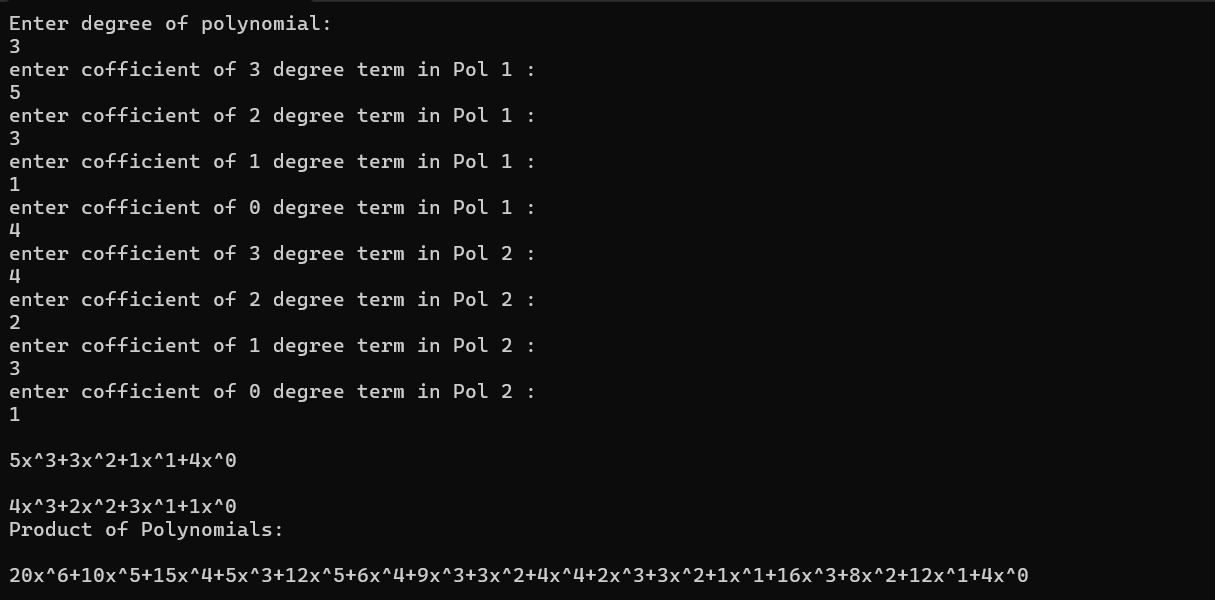
}

8. Write a program to implement multiplication of two polynomials. Each node must

contain the value of the coefficient as well as its power as data components. Take

care of law of exponent multiplication.

Ans :



#include<iostream>

#include<cstring>

using namespace std;

struct node

{

int data;

int degree;

struct node\* next;

};

void print(struct node\*head)

{

struct node\*ptr=head;

cout<<endl;

while(ptr->next!=NULL)

{

cout<<ptr->data<<"x^"<<ptr->degree<<"+";

ptr=ptr->next;

}

cout<<ptr->data<<"x^"<<ptr->degree;

cout<<endl;

}

struct node\* insertatend(struct node\*head, int data, int degree)

{

struct node\*p=new struct node;

struct node\*ptr=head;

p->data=data;

p->degree=degree;

if(ptr==NULL)

{

p->next=NULL;

head=p;

return head;

}

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=p;

p->next=NULL;

return head;

}

struct node\* prodpol(struct node\*h1, struct node\*h2)

{

int degree,data;

struct node\*product=new struct node;

product=NULL;

struct node\*p=h1;

struct node\*q=h2;

while(p!=NULL)

{

while(q!=NULL)

{

degree=p->degree+q->degree;

data=(p->data)\*(q->data);

product=insertatend(product,data,degree) ;

q=q->next;

}

q=h2;

p=p->next;

}

return product;

}

int main()

{

struct node\*h1=NULL;

struct node\*h2= NULL;

int a,k,s;

cout<<"Enter degree of polynomial:\n";

cin>>s;

for(int i=s; i>=0; i--)

{

cout<<"enter cofficient of "<<i<<" degree term in Pol 1 :\n";

cin>>a;

h1=insertatend(h1,a,i);

}

for(int i=s; i>=0; i--)

{

cout<<"enter cofficient of "<<i<<" degree term in Pol 2 :\n";

cin>>a;

h2=insertatend(h2,a,i);

}

print(h1);

print (h2);

cout<<"Product of Polynomials:\n";

struct node\*product=prodpol(h1,h2);

print(product);

}

You can practice questions on Linked lists from Virtual Lab using links:

https://ds1-iiith.vlabs.ac.in/exp/poly-arithmetic/index.html