Name: Swarnabh Paul

Section: Y

Roll no: 19CS8122

Assignment no: 4

Questions attempted: a,b

**Question (a)**

**Contents of Linked List header file:**

**using** **namespace** std;

**class** **node**

{

**public:**

**int** data;

node \*link;

node(**int**,node\*);

};

node::node(**int** x=**0**,node \*l=NULL)

{

data=x;

link=l;

}

**class** **sll**

{

node head;

node\* **createNewNode**(**int**,node\*);

**bool** **isempty**();

**public:**

sll(node \*l);

sll(**const** sll&);

~sll();

**void** **deletesll**();

**void** **insertBeg**(**int**);

**void** **Delete**(**int**);

**bool** **search**(**int**);

**void** **display**();

**int** **size**();

};

sll::sll(node \*l=NULL)

{

head.data=**0**;

head.link=l;

**if**(l!=NULL)

{

**int** cnt=**1**;

node \*t=l;

**while**(t->link!=NULL)

{

t=t->link;

cnt++;

}

head.data=cnt;

}

cout<<"List constructed"<<endl;

}

sll::sll(**const** sll &s)

{

head.data=s.head.data;

head.link=NULL;

node \*t=s.head.link;

**if**(t!=NULL)

{

insertBeg(t->data);

head.data--;

t=t->link;

node \*p=head.link;

**for**(**int** i=**1**;i<s.head.data;i++,t=t->link,p=p->link)

p->link=createNewNode(t->data,NULL);

}

}

**void** sll::deletesll()

{

node \*t;

**for**(**int** i=**0**;i<head.data;i++)

{

t=head.link;

head.link=head.link->link;

**delete** t;

}

head.data=**0**;

}

**bool** sll::isempty()

{

**return** (head.data==**0**);

}

**void** sll::insertBeg(**int** x)

{

head.link=createNewNode(x,head.link);

head.data++;

}

node\* sll::createNewNode(**int** x,node \*l)

{

node \*t=**new** node(x,l);

**return** t;

}

sll::~sll()

{

deletesll();

cout<<"List destroyed"<<endl;

}

**void** sll::Delete(**int** x)

{

**if**(isempty())

{

cout<<"List is empty**\n**";

**return**;

}

node \*p=head.link;

node \*q;

**if**(p->data==x)

{

head.link=p->link;

**delete** p;

head.data--;

}

**else**

{

**while**(p!=NULL&&p->data!=x)

{

q=p;

p=p->link;

}

**if**(p==NULL)

cout<<"No match :: deletion failed**\n**";

**else**

{

q->link=p->link;

**delete** p;

head.data--;

}

}

}

**bool** sll::search(**int** x)

{

node \*t=head.link;

**int** i;

**for**(i=**0**;i<head.data;i++,t=t->link)

**if**(t->data==x)

**return** true;

**return** false;

}

**void** sll::display()

{

node \*t=head.link;

**for**(**int** i=**0**;i<head.data;i++,t=t->link)

{

cout<<t->data<<" --> ";

}

cout<<"||"<<endl;

}

**int** sll::size()

{

**return** head.data;

}

**Code:**

#include<iostream>

#include<stdbool.h>

#include<math.h>

#include<time.h>

#include"MyLinkedList.h"

**using** **namespace** std;

**class** **hashing**

{

sll \*ht;

**int** htsize, mode;

**int** **hashfn**(**int**);

**int** **hashfn2**(**int**);

**public:**

hashing(**int**,**int**);

~hashing();

**bool** **Search**(**int**);

**void** **Insert**(**int**);

**void** **Delete**(**int**);

**void** **Display**();

**double** **TableLoadDistr**();

};

**int** hashing::hashfn(**int** x)

{

**return** (x%htsize);

}

**int** hashing::hashfn2(**int** x)

{

x=abs(x);

**int** s=**0**;

**while**(x)

{

s+=(x%**10**);

x/=**10**;

}

**return** (s%htsize);

}

hashing::hashing(**int** n=**10**,**int** m=**0**)

{

ht=**new** sll[n];

htsize=n;

mode=m%**2**;

}

hashing::~hashing()

{

**for**(**int** i=**0**;i<htsize;i++)

ht[i].deletesll();

**delete** []ht;

}

**bool** hashing::Search(**int** x)

{

**int** index;

**switch**(mode)

{

**case** **0**: index=hashfn(x); **break**;

**case** **1**: index=hashfn2(x); **break**;

}

**return** (ht[index].search(x));

}

**void** hashing::Insert(**int** x)

{

**int** index;

**switch**(mode)

{

**case** **0**: index=hashfn(x); **break**;

**case** **1**: index=hashfn2(x); **break**;

}

ht[index].insertBeg(x);

}

**void** hashing::Delete(**int** x)

{

**int** index;

**switch**(mode)

{

**case** **0**: index=hashfn(x); **break**;

**case** **1**: index=hashfn2(x); **break**;

}

**if**(Search(x))

{

ht[index].Delete(x);

}

**else**

cout<<"Element not found. Deletion not possible."<<endl;

}

**void** hashing::Display()

{

**for**(**int** i=**0**;i<htsize;i++)

ht[i].display();

}

**double** hashing::TableLoadDistr()

{

**int** totalElements=**0**;

**double** deviation=**0.0**, expectedLoad=**100.0**/htsize, bucketLoad;

**for**(**int** i=**0**;i<htsize;i++)

totalElements+=ht[i].size();

cout<<"Expected load in each bucket = "<<expectedLoad<<"%"<<endl;

**for**(**int** i=**0**;i<htsize;i++)

{

bucketLoad=ht[i].size()\***100.0**/totalElements;

cout<<"Load in bucket "<<i+**1**<<" = "<<bucketLoad<<"%"<<endl;

deviation+=fabs(expectedLoad-bucketLoad);

}

**return** deviation;

}

**unsigned** **long** **int** myrand(**unsigned** **long** **int** x)

{

**unsigned** **long** **long** **int** m=**2147483647**, a=**65539**;

**unsigned** **long** **int** r=(x\*a)%m;

**return** r;

}

**int** main()

{

hashing h1, h2(**10**,**1**);

**double** deviation1, deviation2;

**unsigned** **long** **int** seed;

**int** e;

**time\_t** seconds=time(NULL);

seed=seconds;

**for**(**int** i=**0**;i<**100**;i++)

{

seed=myrand(seed);

h1.Insert(seed%**100**+**1**);

h2.Insert(seed%**100**+**1**);

}

cout<<"Displaying hash table:"<<endl;

h1.Display();

deviation1=h1.TableLoadDistr();

cout<<"Total percentage deviation is: "<<deviation1<<endl;

h2.Display();

deviation2=h2.TableLoadDistr();

cout<<"Total percentage deviation is: "<<deviation2<<endl;

**if**(deviation1<deviation2)

{

cout<<"hashfn() is better hash function."<<endl;

}

**else**

{

cout<<"hashfn2() is better hash function."<<endl;

}

cout<<"Enter value to search for: ";

cin>>e;

**if**(h1.Search(e))

cout<<"Search successful. Element found.**\n**";

**else**

cout<<"Search unsuccessful. Element not found.**\n**";

cout<<"Enter element to delete: ";

cin>>e;

h1.Delete(e);

cout<<"Displaying hash table:"<<endl;

h1.Display();

**return** **0**;

}

**Output:**

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

List constructed

Displaying hash table:

30 --> 60 --> 60 --> 50 --> 100 --> 40 --> 10 --> 10 --> 100 --> ||

61 --> 41 --> 91 --> 51 --> 61 --> ||

62 --> 92 --> 12 --> 12 --> 92 --> 42 --> 42 --> 12 --> 62 --> 12 --> ||

83 --> 63 --> 3 --> 53 --> 53 --> 73 --> 43 --> 13 --> 53 --> 23 --> 83 --> ||

34 --> 84 --> 4 --> 34 --> 54 --> 34 --> 94 --> 4 --> 44 --> 84 --> 54 --> 54 --> 74 --> 54 --> 4 --> 94 --> 34 --> ||

35 --> 5 --> 45 --> 35 --> 55 --> 65 --> 15 --> 85 --> 45 --> 95 --> ||

56 --> 76 --> 66 --> 36 --> 96 --> 86 --> 36 --> 66 --> 66 --> 56 --> 56 --> 36 --> ||

47 --> 67 --> 47 --> 7 --> 57 --> 87 --> 17 --> 87 --> ||

78 --> 78 --> 58 --> 78 --> 88 --> 68 --> 28 --> ||

79 --> 59 --> 39 --> 39 --> 39 --> 49 --> 89 --> 59 --> 39 --> 19 --> 59 --> ||

Expected load in each bucket = 10%

Load in bucket 1 = 9%

Load in bucket 2 = 5%

Load in bucket 3 = 10%

Load in bucket 4 = 11%

Load in bucket 5 = 17%

Load in bucket 6 = 10%

Load in bucket 7 = 12%

Load in bucket 8 = 8%

Load in bucket 9 = 7%

Load in bucket 10 = 11%

Total percentage deviation is: 22

73 --> 55 --> 91 --> 19 --> 28 --> ||

56 --> 47 --> 83 --> 47 --> 92 --> 92 --> 65 --> 100 --> 10 --> 56 --> 10 --> 100 --> 56 --> 74 --> 83 --> ||

84 --> 66 --> 39 --> 39 --> 39 --> 57 --> 39 --> 66 --> 84 --> 66 --> ||

67 --> 30 --> 76 --> 58 --> 49 --> 3 --> 12 --> 12 --> 94 --> 12 --> 12 --> 85 --> 94 --> ||

59 --> 4 --> 59 --> 86 --> 4 --> 40 --> 13 --> 59 --> 68 --> 4 --> 95 --> ||

78 --> 78 --> 5 --> 78 --> 87 --> 96 --> 41 --> 50 --> 23 --> 87 --> ||

79 --> 60 --> 60 --> 42 --> 51 --> 42 --> 15 --> 88 --> ||

34 --> 61 --> 34 --> 7 --> 89 --> 34 --> 43 --> 61 --> 34 --> ||

35 --> 62 --> 53 --> 53 --> 35 --> 44 --> 17 --> 62 --> 53 --> ||

36 --> 63 --> 45 --> 54 --> 36 --> 54 --> 54 --> 45 --> 54 --> 36 --> ||

Expected load in each bucket = 10%

Load in bucket 1 = 5%

Load in bucket 2 = 15%

Load in bucket 3 = 10%

Load in bucket 4 = 13%

Load in bucket 5 = 11%

Load in bucket 6 = 10%

Load in bucket 7 = 8%

Load in bucket 8 = 9%

Load in bucket 9 = 9%

Load in bucket 10 = 10%

Total percentage deviation is: 18

hashfn2() is better hash function.

Enter value to search for: 40

Search successful. Element found.

Enter element to delete: 40

Displaying hash table:

30 --> 60 --> 60 --> 50 --> 100 --> 10 --> 10 --> 100 --> ||

61 --> 41 --> 91 --> 51 --> 61 --> ||

62 --> 92 --> 12 --> 12 --> 92 --> 42 --> 42 --> 12 --> 62 --> 12 --> ||

83 --> 63 --> 3 --> 53 --> 53 --> 73 --> 43 --> 13 --> 53 --> 23 --> 83 --> ||

34 --> 84 --> 4 --> 34 --> 54 --> 34 --> 94 --> 4 --> 44 --> 84 --> 54 --> 54 --> 74 --> 54 --> 4 --> 94 --> 34 --> ||

35 --> 5 --> 45 --> 35 --> 55 --> 65 --> 15 --> 85 --> 45 --> 95 --> ||

56 --> 76 --> 66 --> 36 --> 96 --> 86 --> 36 --> 66 --> 66 --> 56 --> 56 --> 36 --> ||

47 --> 67 --> 47 --> 7 --> 57 --> 87 --> 17 --> 87 --> ||

78 --> 78 --> 58 --> 78 --> 88 --> 68 --> 28 --> ||

79 --> 59 --> 39 --> 39 --> 39 --> 49 --> 89 --> 59 --> 39 --> 19 --> 59 --> ||

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

List destroyed

**Question (b)**

**Code:**

#include<iostream>

#include<stdbool.h>

#include<math.h>

**using** **namespace** std;

**class** **node**

{

**public:**

**int** exp, coeff;

node \*link;

node(**int**,**int**,node\*);

};

node::node(**int** c=**0**,**int** e=**0**,node \*l=NULL)

{

coeff=c;

exp=e;

link=l;

}

**class** **polynomial**

{

node head;

**bool** **isempty**();

node \***createNewNode**(**int**,**int**,node\*);

**public:**

polynomial(node\*);

polynomial(**const** polynomial&);

~polynomial();

**void** **DeletePoly**();

**void** **InsertTerm**(**int**,**int**);

**void** **DeleteTerm**(**int**,**int**);

**void** **DisplayPoly**();

**double** **EvalPoly**(**double**);

polynomial\* **AddPoly**(**const** polynomial &p);

};

**bool** polynomial::isempty()

{

**return** (head.coeff==**0**);

}

node\* polynomial::createNewNode(**int** c,**int** e,node \*l)

{

node \*t=**new** node(c,e,l);

**return** t;

}

polynomial::polynomial(node \*l=NULL)

{

head.coeff=**0**;

head.exp=**0**;

head.link=l;

**if**(l!=NULL)

{

**int** cnt=**0**, Max=l->exp;

node \*t=l;

**while**(t=NULL)

{

t=t->link;

cnt++;

**if**(t->exp>Max)

Max=t->exp;

}

head.coeff=cnt;

head.exp=Max;

}

cout<<"Polynomial constructed"<<endl;

}

polynomial::polynomial(**const** polynomial &p)

{

head.coeff=p.head.coeff;

head.exp=p.head.exp;

head.link=NULL;

node \*t=p.head.link;

**if**(t!=NULL)

{

head.link=createNewNode(t->coeff,t->exp,head.link);

t=t->link;

node \*q=head.link;

**for**(**int** i=**1**;i<p.head.coeff;i++,q=q->link,t=t->link)

q->link=createNewNode(t->coeff,t->exp,NULL);

}

}

polynomial::~polynomial()

{

DeletePoly();

cout<<"Polynomial destroyed"<<endl;

}

**void** polynomial::DeletePoly()

{

node \*t;

**for**(**int** i=**0**;i<head.coeff;i++)

{

t=head.link;

head.link=head.link->link;

**delete** t;

}

head.coeff=**0**;

}

**void** polynomial::InsertTerm(**int** c,**int** e)

{

**if**(isempty())

{

head.link=createNewNode(c,e,head.link);

}

**else** **if**(head.link->exp>e)

{

head.link=createNewNode(c,e,head.link);

}

**else**

{

node \*p=head.link;

**while**(p->link!=NULL&&p->link->exp<e)

p=p->link;

p->link=createNewNode(c,e,p->link);

}

head.coeff++;

}

**void** polynomial::DeleteTerm(**int** c,**int** e)

{

**if**(isempty())

{

cout<<"Polynomial is empty**\n**";

**return**;

}

node \*p=head.link;

node \*q;

**if**(p->coeff==c&&p->exp==e)

{

head.link=p->link;

**delete** p;

head.coeff--;

}

**else**

{

**while**(p!=NULL&&(p->coeff!=c||p->exp!=e))

{

q=p;

p=p->link;

}

**if**(p==NULL)

cout<<"No match :: deletion failed**\n**";

**else**

{

q->link=p->link;

**delete** p;

head.coeff--;

}

}

}

**void** polynomial::DisplayPoly()

{

**if**(head.coeff==**0**)

{

cout<<**0**<<endl;

**return**;

}

node \*t=head.link;

cout<<t->coeff<<"x^"<<t->exp;

t=t->link;

**for**(**int** i=**1**;i<head.coeff;i++,t=t->link)

{

**if**(t->coeff>=**0**)

cout<<"+";;

cout<<t->coeff<<"x^"<<t->exp;

}

cout<<endl;

}

**double** polynomial::EvalPoly(**double** x)

{

**double** result=**0.0**;

node \*t=head.link;

**for**(**int** i=**0**;i<head.coeff;i++,t=t->link)

result+=((t->coeff)\*pow(x,t->exp));

**return** result;

}

polynomial\* polynomial::AddPoly(**const** polynomial &p)

{

**int** i=**0**;

polynomial \*t=**new** polynomial;

node \*a=head.link;

node \*b=p.head.link;

node \*c;

t->head.coeff=**0**;

t->head.exp=**0**;

t->head.link=NULL;

**while**(a!=NULL&&b!=NULL)

{

**if**(a->exp<b->exp)

{

**if**(t->head.coeff==**0**)

{

t->head.link=createNewNode(a->coeff,a->exp,t->head.link);

c=t->head.link;

}

**else**

{

c->link=createNewNode(a->coeff,a->exp,c->link);

c=c->link;

}

(t->head.coeff)++;

a=a->link;

}

**else** **if**(a->exp>b->exp)

{

**if**(t->head.coeff==**0**)

{

t->head.link=createNewNode(b->coeff,b->exp,t->head.link);

c=t->head.link;

}

**else**

{

c->link=createNewNode(b->coeff,b->exp,c->link);

c=c->link;

}

(t->head.coeff)++;

b=b->link;

}

**else**

{

**if**(t->head.coeff==**0**)

{

t->head.link=createNewNode(a->coeff+b->coeff,a->exp,t->head.link);

c=t->head.link;

}

**else**

{

c->link=createNewNode(a->coeff+b->coeff,a->exp,c->link);

c=c->link;

}

(t->head.coeff)++;

a=a->link;

b=b->link;

}

}

**while**(a!=NULL)

{

**if**(t->head.coeff==**0**)

{

t->head.link=createNewNode(a->coeff,a->exp,t->head.link);

c=t->head.link;

}

**else**

{

c->link=createNewNode(a->coeff,a->exp,c->link);

c=c->link;

}

(t->head.coeff)++;

a=a->link;

}

**while**(b!=NULL)

{

**if**(t->head.coeff==**0**)

{

t->head.link=createNewNode(b->coeff,b->exp,t->head.link);

c=t->head.link;

}

**else**

{

c->link=createNewNode(b->coeff,b->exp,c->link);

c=c->link;

}

(t->head.coeff)++;

b=b->link;

}

**return** t;

}

**int** main()

{

polynomial p;

p.InsertTerm(-**4**,**3**);

p.InsertTerm(**2**,**2**);

p.InsertTerm(**6**,**0**);

p.InsertTerm(-**7**,**1**);

p.DisplayPoly();

polynomial q=p;

p.DeleteTerm(**6**,**0**);

p.DisplayPoly();

q.DisplayPoly();

cout<<q.EvalPoly(-**2.5**)<<endl;

polynomial a,b;

a.InsertTerm(**5**,**0**);

a.InsertTerm(-**1**,**1**);

a.InsertTerm(**1**,**2**);

a.InsertTerm(**4**,**3**);

cout<<"a = ";

a.DisplayPoly();

b.InsertTerm(**2**,**1**);

b.InsertTerm(**6**,**2**);

b.InsertTerm(-**10**,**0**);

cout<<"b = ";

b.DisplayPoly();

polynomial \*t=b.AddPoly(a);

cout<<"a+b = ";

t->DisplayPoly();

**delete** t;

**return** **0**;

}

**Output:**

Polynomial constructed

6x^0-7x^1+2x^2-4x^3

-7x^1+2x^2-4x^3

6x^0-7x^1+2x^2-4x^3

98.5

Polynomial constructed

Polynomial constructed

a = 5x^0-1x^1+1x^2+4x^3

b = -10x^0+2x^1+6x^2

Polynomial constructed

a+b = -5x^0+1x^1+7x^2+4x^3

Polynomial destroyed

Polynomial destroyed

Polynomial destroyed

Polynomial destroyed

Polynomial destroyed