# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 8 |
| Roll No | 1333 |
| Unit | 5 |
| Program | Singly Linked list |

**Source Code:**

#include<iostream> #include<conio.h> using namespace std;

/\* 1. Node Template \*/ class Node

{

public:

int data; Node \*next;

};

/\* 2. List Template \*/ class List

{

Node \*head; public:

List()

{

head = NULL;

}

void Insert(int x); void Display(); void Length(); void Search(int x);

void Remove(int x); void Reverse();

};

/\* 3. Functions \*/

//1. Insert

void List :: Insert(int x)

# MET Institute of Computer Science

{

//create a node

Node \*t = new Node(); t->data = x;

t->next = NULL;

//if list is empty if(head == NULL)

{

head = t; return;

}

//traverse till the last node and attach t Node \*tmp = head;

while(tmp->next != NULL)

{

tmp = tmp->next;

}

//Attach t to tmp tmp->next = t;

}

//2. Display

void List :: Display()

{

Node \*tmp = head; while(tmp!=NULL)

{

//cout << "Node at: " << tmp;

//cout << " Data is: " << tmp->data;

//cout << " Next Node at: " << tmp->next;

//cout << endl;

cout << tmp->data << " -> "; tmp = tmp->next;

}

cout << "NULL";

}

//3. length

# MET Institute of Computer Science

void List :: Length()

{

Node \*tmp = head; int cnt = 0; while(tmp!=NULL)

{

cnt++; //cnt = cnt + 1; tmp = tmp->next;

}

cout << "Number of nodes: " << cnt;

}

//4. Search

void List :: Search(int x)

{

Node \*tmp = head; int flag = 0; while(tmp!=NULL)

{

if(tmp->data == x)

{

flag = 1; break;

}

tmp = tmp->next;

}

if(flag == 1)

{

cout << x << " is found in the list";

}

else

{

cout << x << " is not found in the list";

}

}

//5. remove

void List :: Remove(int x)

# MET Institute of Computer Science

{

// step 1: Empty List - RETURN CONTROL if(head == NULL)

{

cout << "Empty List"; return;

}

// step 2: Search for x Node \*tmp = head; int flag = 0;

Node \*prev = NULL; while(tmp!=NULL)

{

if(tmp->data == x)

{

flag = 1; break;

}

prev = tmp;

tmp = tmp->next;

}

// step 3: unsuccessful search - RETURN CONTROL if(flag == 0)

{

cout << x << " is not found in the list"; return;

}

// step 4: Search is successful

// step 4a: Single node deletion

if(tmp == head && tmp->next == NULL)

{

head = NULL;

}

else if(tmp == head) //step 4b: Head node deletion

{

head = tmp->next;

# MET Institute of Computer Science

}

else if(tmp->next == NULL) //step 4c: Tail node deletion

{

prev->next = NULL;

}

else //step 4d: Any other node in the middle

{

prev->next = tmp->next;

}

//step 5: delete tmp delete tmp;

}

//6. Reverse

void List :: Reverse()

{

if(head == NULL)

{

cout << "Empty List"; return;

}

Node \*tmp = head; Node \*revHead = NULL; while(tmp != NULL)

{

//create a new node Node \*r = new Node(); r->data = tmp->data;

r->next = revHead; //for the 1st node revHead is NULL

//update revHead revHead = r;

//advance tmp to the next node tmp = tmp->next;

}

//update head to revHead head = revHead;

cout << endl << "Reversed List: ";

# MET Institute of Computer Science

List :: Display();

}

/\* 4. Menu \*/ int main()

{

int ch; int num; List s; while(1)

{

system("cls");

cout << "\*\*\* Singly Lined List Menu \*\*\*" << endl << endl; cout << "1. Insert the node" << endl;

cout << "2. Display the list" << endl; cout << "3. Length of the list" << endl; cout << "4. Search for a Node" << endl; cout << "5. Remove a Node" << endl; cout << "6. Reverse the list" << endl; cout << "7. Exit" << endl << endl;

cout << "Enter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter the element: "; cin >> num;

s.Insert(num); getch(); break;

case 2: s.Display();

getch(); break; case 3: s.Length();

getch();

# MET Institute of Computer Science

break; case 4:

cout << "Enter the element: "; cin >> num;

s.Search(num); getch();

break; case 5:

cout << "Enter the element: "; cin >> num;

s.Remove(num); s.Display();

getch(); break; case 6:

s.Reverse();

getch(); break; case 7:

exit(1); default:

cout << "Incorrect option"; getch();

}//end of switch

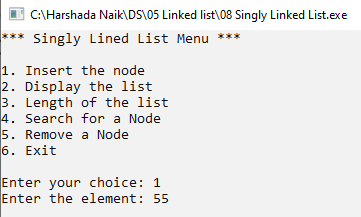
}//end of while

}//end of main

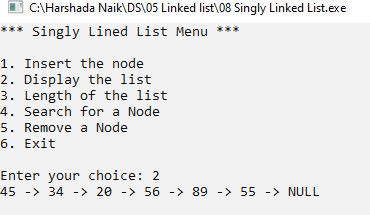
**Output:**

**Insert an element:**

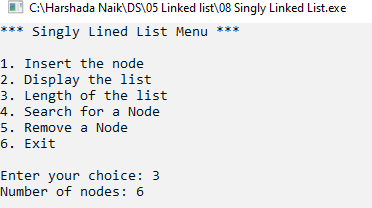
# MET Institute of Computer Science



**Display a list:**

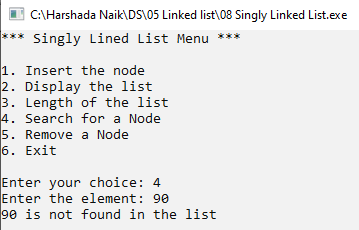


**Length of the List:**

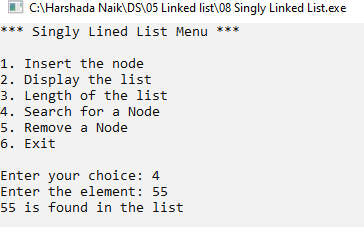


**Search for unsuccessful node**

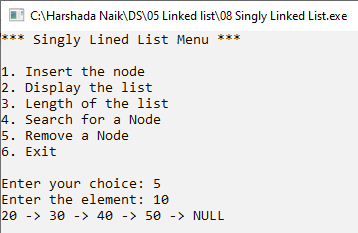
# MET Institute of Computer Science



**Search for successful node**

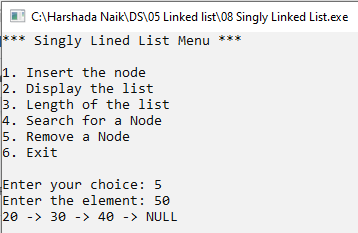


**Head node deletion**

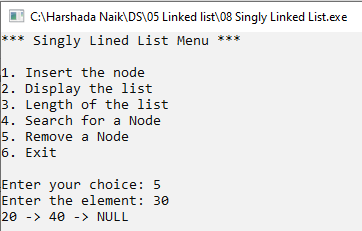


**Tail node deletion**

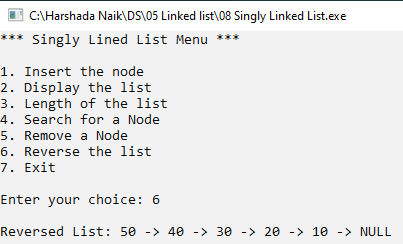
# MET Institute of Computer Science



**Any other deletion of node in middle**



**Reverse a List:**



# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 9 |
| Roll No | 1333 |
| Unit | 5 |
| Program | Circular Linked list |

**Source Code:**

#include<iostream>

#include<conio.h> using namespace std;

/\* 1. Node Template \*/ class CNode

{

public:

int data; CNode \*next;

};

/\* 2. List Template \*/ class CList

{

CNode \*first; CNode \*last; public:

CList()

{

first = NULL; last = NULL;

}

void Insert(int x); void Display(); void Length(); void Search(int x);

void Remove(int x);

};

/\* 3. Functions \*/

// 1. Insert

void CList :: Insert(int x)

# MET Institute of Computer Science

{

//create a Node

CNode \*t = new CNode(); t->data = x;

t->next = NULL;

//if list is empty

if(first == NULL && last == NULL)

{

first = t; last = t;

last->next = first;

}

else // Attach t to last, shift last, make it circular

{

last->next = t; last = t;

last->next = first;

}

}

// 2. Display

void CList :: Display()

{

if(first == NULL)

{

cout << "Empty List"; return;

}

CNode \*tmp = first; do

{

cout << tmp->data << " -> "; tmp = tmp->next;

}

while(tmp != first); //tmp has not come back to first cout << "Back to first";

}

# MET Institute of Computer Science

// 3. Length

void CList :: Length()

{

if(first == NULL)

{

cout << "Empty List"; return;

}

CNode \*tmp = first; int cnt = 0;

do

{

cnt++;

tmp = tmp->next;

}

while(tmp != first);

cout << "Number of nodes: " << cnt;

}

// 4. Search

void CList :: Search(int x)

{

if(first == NULL)

{

cout << "Empty List"; return;

}

CNode \*tmp = first; int flag = 0;

do

{

if(tmp->data == x)

{

flag = 1; break;

# MET Institute of Computer Science

}

tmp = tmp->next;

}

while(tmp != first);

{

if(flag == 1)

{

cout << x << " is found in the list";

}

else

{

cout << x << " is not found in the list";

}

}

}

//5. delete

void CList :: Remove(int x)

{

//step 1- empty list - return control if(first == NULL)

{

cout << "Empty List"; return;

}

//step 2 - search for x CNode \*tmp = first; CNode \*prev = NULL; int flag = 0;

do

{

if(tmp->data == x)

{

flag = 1; break;

}

prev = tmp;

# MET Institute of Computer Science

tmp = tmp->next;

}

while( tmp!= first);

//step 3 - search unsuccessful - RETURN CONTROL if(flag == 0)

{

cout << x << " is not found"; return;

}

// step 4: Search is successful

// step 4a: Single node deletion if(first == last)

{

first = NULL; last = NULL;

}

else if(tmp == first) // step 4b: First node Deletion

{

first = tmp->next; last->next = first;

}

else if(tmp == last) // step 4c: Last node Deletion

{

last = prev;

last->next = first;

}

else //step 4d: if any other node in the middle

{

prev->next = tmp->next;

}

//step 5: delete tmp delete tmp;

CList :: Display();

}

/\* 4. Menu \*/ int main()

# MET Institute of Computer Science

{

int ch; int num; CList c; while(1)

{

system("cls");

cout << "\*\*\*Circular Linked List\*\*\*\n\n"; cout << "1. Insert a node" << endl; cout << "2. Displya a list" << endl;

cout << "3. Length of the list" << endl; cout << "4. Search for a node" << endl; cout << "5. Remove a node" << endl; cout << "6. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter an element: "; cin >> num;

c.Insert(num); getch(); break;

case 2:

c.Display();

getch(); break; case 3:

c.Length();

getch(); break; case 4:

cout << "Enter an element: "; cin >> num;

c.Search(num);

# MET Institute of Computer Science

getch(); break; case 5:

cout << "Enter an element:"; cin >> num; c.Remove(num);

getch(); break; case 6:

exit(1); default:

cout << "Incorrect Option"; getch();

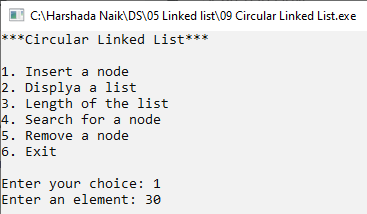
}//end of switch

}//end of while

}//end of main

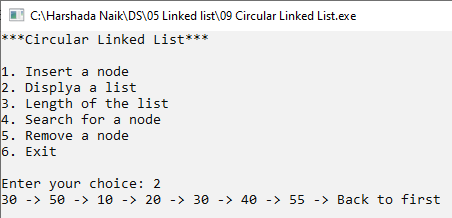
**Output:**

**Insert an element:**

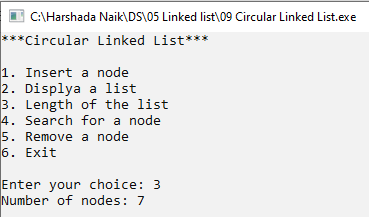


**Display a list:**

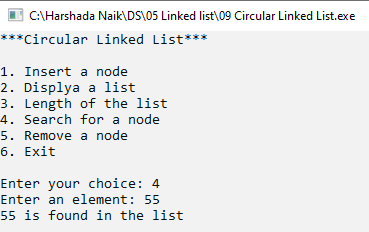
# MET Institute of Computer Science



**Length of the list:**

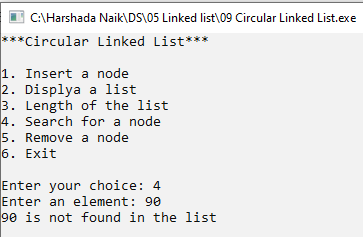


**Search for a successful node:**

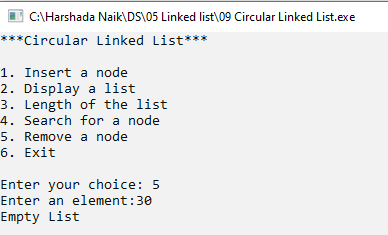


**Search for a unsuccessful node:**

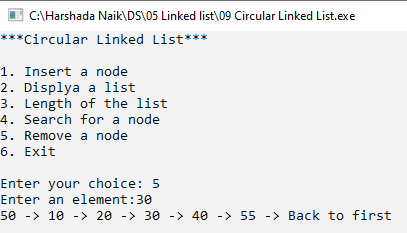
# MET Institute of Computer Science



**Single node deletion**

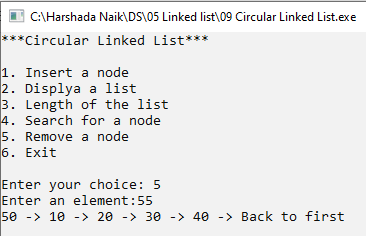


**Head node deletion**

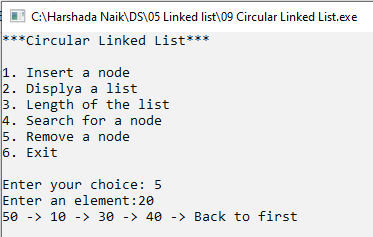


**Tail node deletion**

# MET Institute of Computer Science



**Any other node deletion in middle**



# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 10 |
| Roll No | 1333 |
| Unit | 5 |
| Program | Doubly Linked list |

**Source Code: #include<iostream>**

#include<conio.h> using namespace std;

/\*1. Node Template \*/ class DNode

{

public:

int data; DNode \*right; DNode \*left;

};

/\*2. List Template \*/ class DList

{

DNode \*head; DNode \*tail; public:

DList()

{

head=NULL; tail=NULL;

}

void Insert(int x); void Display(); void Length(); void Search(int x);

void Remove(int x);

};

/\*3. Function \*/

# MET Institute of Computer Science

// Insertion

void DList :: Insert(int x)

{

//Make a new node DNode \*t = new DNode(); t->data = x;

t->right = NULL; t->left = NULL;

//Head Node if(head == NULL)

{

head = t; tail = t;

}

//Attach t to tail and shift tail else

{

tail->right = t; t->left = tail; tail = t;

}

}

//Display

void DList :: Display(){ if(head == tail)

{

cout << "Empty List"; return;

}

cout << "Forward Direction: "; DNode \*tmp = head; while(tmp!=NULL)

{

cout << tmp->data << " <-> "; tmp = tmp->right;

}

# MET Institute of Computer Science

cout << "End Of List" << endl; cout << "Reverse Direction: "; tmp = tail;

while(tmp!=NULL)

{

cout << tmp->data << " <-> "; tmp = tmp->left;

}

cout << "Start Of List" << endl;

}

//Length

void DList :: Length()

{

int cnt = 0;

DNode \*tmp = head; if(head == NULL)

{

cout << "Empty List"; return;

}

while(tmp!=NULL)

{

cnt++;

tmp = tmp->right;

}

cout << "Number Of Nodes: " << cnt;

}

//Search

void DList :: Search(int x)

{

DNode \*tmp = head; int flag = 0; while(tmp!=NULL)

{

if(tmp->data == x )

{

# MET Institute of Computer Science

flag = 1; break;

}

tmp = tmp->right;

}

if(flag == 1)

{

cout << x << " is found in the List.";

}

else

{

cout << x << " is not found in the List.";

}

}

//Delete

void DList :: Remove(int x)

{

//1. Empty list - Return Control if(head==NULL)

{

cout << "Empty List"; return;

}

//2. Search for x DNode \*tmp = head; DNode \*prev = NULL; int flag = 0; while(tmp!=NULL)

{

if(tmp->data == x )

{

flag = 1; break;

}

prev = tmp;

tmp = tmp->right;

# MET Institute of Computer Science

}

//3. Search unsuccessful - Return Control if(flag == 0)

{

cout << x << " is not found"; return;

}

//4. Search Successful

//4a. Single Node Deletion if(head == tail)

{

head = NULL; tail = NULL;

}

//4b. Head Node Deletion else if(tmp == head)

{

head = tmp->right; head->left = NULL;

}

//4c. Tail Node Deletion else if(tmp == tail)

{

tail = prev;

tail->right = NULL;

}

//4d. Any other node in the middle else

{

prev->right = tmp->right; (tmp->right)->left = prev;

}

//5. Delete tmp delete tmp; DList::Display();

}

# MET Institute of Computer Science

/\* 4. Menu Interface \*/ int main()

{

int ch; int num; DList d; while(1)

{

system("cls");

cout << "\*\*\* Doubly Linked List \*\*\* \n\n"; cout << "1. Insert a node" << endl;

cout << "2. Display the list" << endl; cout << "3. Length of the list" << endl; cout << "4. Search a node" << endl; cout << "5. Remove a node" << endl; cout << "6. Exit" << endl << endl;

cout << "Enter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter an Element: "; cin >> num;

d.Insert(num); getch(); break;

case 2: d.Display();

getch(); break; case 3: d.Length();

getch(); break; case 4:

cout << "Enter an Element: ";

# MET Institute of Computer Science

cin >> num; d.Search(num); getch();

break; case 5:

cout << "Enter an Element: "; cin >> num; d.Remove(num);

getch(); break; case 6:

exit(1); default:

cout << "Incorrect option" << endl; getch();

break;

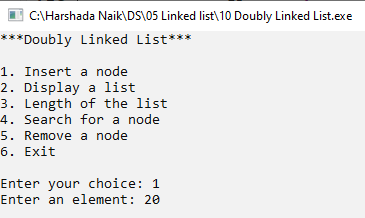
}//end of switch

}//end of while

}//end of main

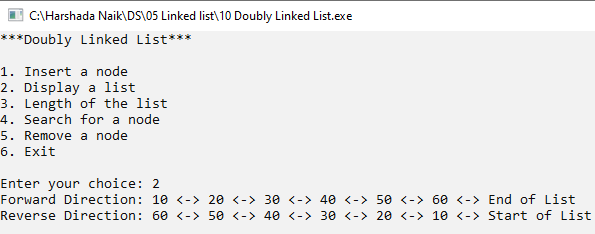
**Output:**

**Insert an element**

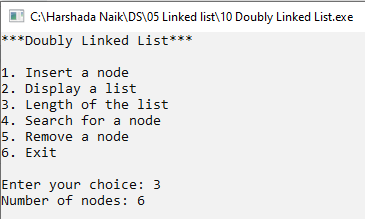


**Display a list:**

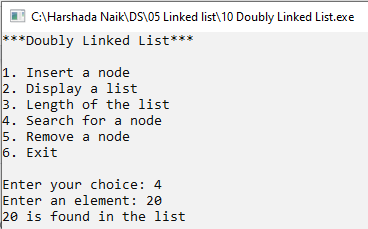
# MET Institute of Computer Science



**Length of the list:**

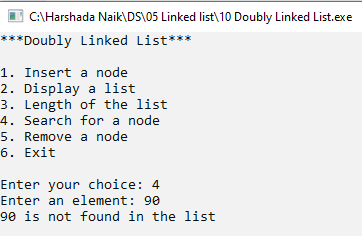


**Search for a successful node:**

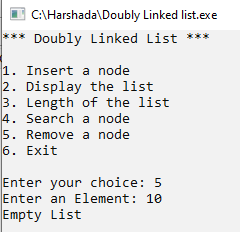


**Search for a unsuccessful node**

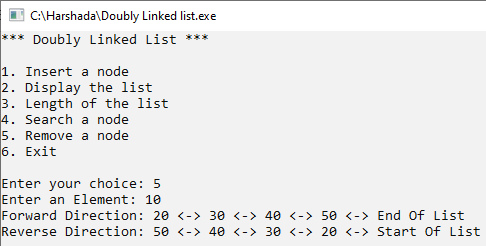
# MET Institute of Computer Science



**Single node deletion**



**Head node deletion**



**Tail Node deletion**

# MET Institute of Computer Science

|  |  |  |
| --- | --- | --- |
|  | |  |
| **Any other node deletion in between** | | |
|  |  | |

**MET Institute of Computer Science**

|  |  |
| --- | --- |
| Program No | 11 |
| Roll No | 1333 |
| Unit | 5 |
| Program | polynomial Addition |

**Source Code: #include<iostream>**

#include<conio.h> using namespace std;

/\* 1. Node Template \*/ class PNode

{

public:

int coeff; int exp;

PNode \*next;

};

/\* 2. List Template \*/ class PList

{

PNode \*head; public:

PList()

{

head = NULL;

}

void Insert(int c, int e); void Display();

void Add(PList p1, PList p2);

};

/\* 3. Function \*/

//Insert

void PList :: Insert(int c, int e)

{

//Make a new Node

# MET Institute of Computer Science

PNode \*t = new PNode(); t->coeff = c;

t->exp = e;

t->next = NULL;

//First Node if(head == NULL)

{

head = t; return;

}

//Traverse and insert in order PNode \*tmp = head;

PNode \*prev = NULL;

while(tmp != NULL && tmp->exp>e)

{

prev = tmp;

tmp = tmp->next;

}

//Head Node if(tmp == head)

{

t->next = tmp; head = t;

}

//Tail Node

else if(tmp == NULL)

{

prev->next = t;

}

//Any node in the middle else

{

t->next = tmp; prev->next = t;

}

}

# MET Institute of Computer Science

//Display

void PList :: Display()

{

if(head == NULL)

{

cout << "Polynomial is empty"; return;

}

PNode \*tmp = head; while(tmp!=NULL)

{

cout << tmp->coeff;

cout << "x^" << tmp->exp; if(tmp->next != NULL)

{

if((tmp->next)->coeff > 0)

{

cout << " + ";

}

else

{

cout << " ";

}

}

tmp=tmp->next;

}

}

//Add

void PList :: Add(PList p1, PList p2)

{

PNode \*P=p1.head; PNode \*Q=p2.head; int c,e;

while(P!=NULL && Q!=NULL)

{

if(P->exp == Q->exp)

# MET Institute of Computer Science

{

c=P->coeff+Q->coeff; if(c != 0)

{

e = P->exp; Insert(c,e);

}

P=P->next; Q=Q->next;

}//end of P->exp == Q->exp else if(P->exp > Q->exp)

{

c=P->exp; e=P->exp; Insert(c,e); P=P->next;

}//end of P->exp > Q->exp else

{

c=Q->coeff; e=Q->exp; Insert(c,e); Q=Q->next;

}//end of Q->exp > P->exp

}//end of while P&&Q != NULL while(P!=NULL)

{

c=P->coeff; e=P->exp; Insert(c,e); P=P->next;

}//end of while P is not NULL but Q is. while(Q != NULL)

{

c=Q->coeff; e=Q->exp;

# MET Institute of Computer Science

Insert(c,e); Q=Q->next;

}//end of while Q is not NULL but P is.

}

/\* 4. Menu \*/ int main()

{

int ch, co, ex; PList p1,p2,p3; while(1)

{

system("cls");

cout << "\*\*\* Polynomial Function \*\*\*\n\n";

cout << "1. Read the first polynomial" << endl; cout << "2. Read the second polynomial" << endl; cout << "3. Display the first polynomial" << endl;

cout << "4. Display the second polynomial" << endl; cout << "5. Add the polynomials" << endl;

cout << "6. Exit" << endl << endl; cout << "Enter your choice: ";

cin >> ch; switch(ch)

{

case 1:

cout << "Enter the coefficient: "; cin >> co;

cout << "Enter the exponent: "; cin >> ex;

p1.Insert(co,ex); getch();

break; case 2:

cout << "Enter the coefficient: "; cin >> co;

cout << "Enter the exponent: "; cin >> ex;

# MET Institute of Computer Science

p2.Insert(co,ex); getch();

break; case 3:

p1.Display();

getch(); break; case 4:

p2.Display();

getch(); break; case 5:

p3.Add(p1,p2);

p3.Display();

getch(); break; case 6:

exit(1); default:

cout << "Incorrect option" << endl; getch();

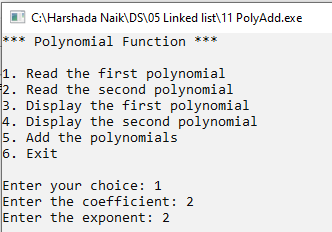
break;

}//end of switch

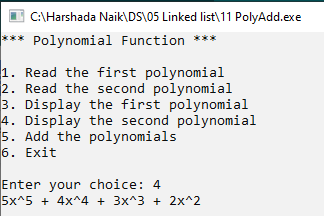
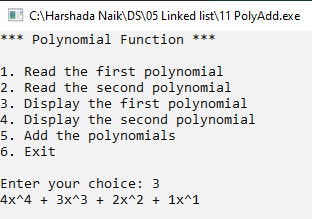
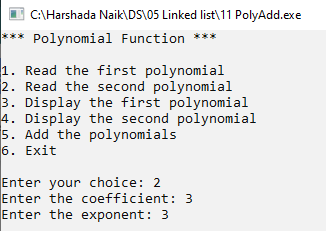
}//end of while

}//end of main

**Output:**



# MET Institute of Computer Science



**MET Institute of Computer Science**

