**M. Ali. Arslan**

**19F-0348**

**Lab-02**

**Task # 01**

#include<iostream>

using namespace std;

class node

{

public:

int data;

node\* next;

node(int d)

{

data = d;

next = NULL;

}

};

void insert(node\*& head, int d)

{

node\* n = new node(d);

n->next = head;

head = n;

}

void input(node\*& head)

{

int d;

cin >> d;

while (d != -1) //-1 marks termination of linked list

{

insert(head, d);

cin >> d;

}

}

void printlength(node\* head)

{

while (head != NULL)

{

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

head = head->next;

}

cout << endl;

}

int length(node\* temp)

{

int count = 0;

while (temp != NULL)

{

count++;

temp = temp->next;

}

return count;

}

int main()

{

int choice;

cout << "1) Length of Link List"<<endl;

cout << "2) Print the whole Link List" << endl;

cout << "Choice: ";

cin >> choice;

switch (choice)

{

case 1:

{

int l;

cout << "Enter elements of link list & enter -1 at the last to end the list:" << endl;

node\* head = NULL;

input(head);

printlength(head);

l = length(head);

cout << "Length of Link List is: " << l;

break;

}

case 2:

{

cout << "Enter elements of link list & enter -1 at the last to end the list:" << endl;

node\* head = NULL;

input(head);

cout << "The link list is: ";

printlength(head);

}

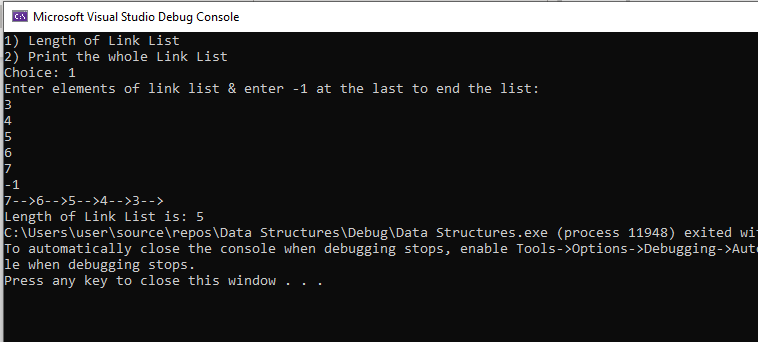
default:

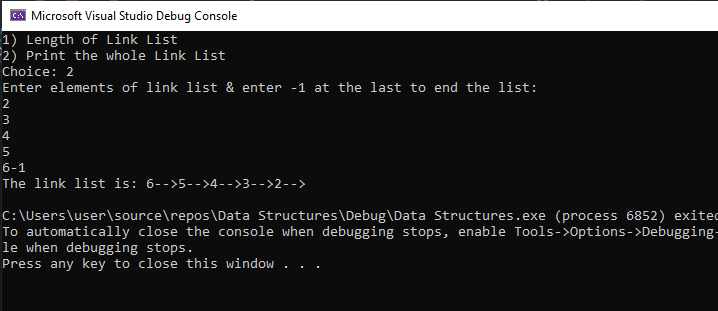
break;

}

return 0;

}





**Task # 02**

#include<iostream>

using namespace std;

struct node

{

node\* next;

int data;

};

class list

{

node\* head;

node\* tail;

int size;

public:

list()

{

head = NULL;

tail = NULL;

size = 0;

}

void insertnode(int n)

{

node\* Node1 = new node;

Node1->data = n;

Node1->next = NULL;

size++;

if (head == NULL)

{

head = Node1;

tail = Node1;

}

else

tail->next = Node1;

tail = Node1;

}

int count()

{

return size;

}

void print()

{

int counter = 0;

node\* temp;

if (head == NULL)

{

cout << "List is empty!" << endl;

}

else

{

temp = head;

cout << "\nOutput is: ";

while (temp != NULL)

{

counter++;

if (counter % 2 != 0)

{

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

}

temp = temp->next;

}

}

}

void print1()

{

int counter = 0;

node\* temp;

if (head == NULL)

{

cout << "This list is Empty!" << endl;

}

else

{

temp = head;

while (temp != NULL)

{

counter++;

if (counter % 2 == 0)

{

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

}

temp = temp->next;

}

}

}

};

int main()

{

list obj;

int n;

cout << "Enter Number of nodes: ";

cin >> n;

int num;

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

{

cout << "Enter value of node " << i+1 << ": ";

cin >> num;

obj.insertnode(num);

}

int x = obj.count();

obj.print();

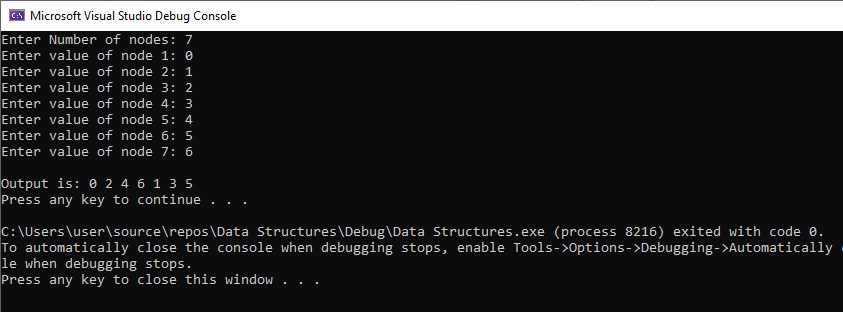
obj.print1();

cout << endl;

system("pause");

return 0;

}



**Task # 03**

#include <iostream>

using namespace std;

void insertsort(int arr[], int first, int size);

int main()

{

int arr[] = { 15,1,-3,55,10,0};

int size = sizeof(arr) / sizeof(arr[0]);

insertsort(arr, 1, size - 1);

cout << "The input array is: 15 1 -3 55 10 0" << endl;

cout << "Array after insertion sort: ";

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

{

cout << arr[i] << " ";

}

cout << endl;

system("pause");

return 0;

}

void insertsort(int arr[], int first, int size)

{

int index = arr[first];

int i = first;

while (i > 0 && arr[i - 1] > index)

{

arr[i] = arr[i - 1];

i--;

}

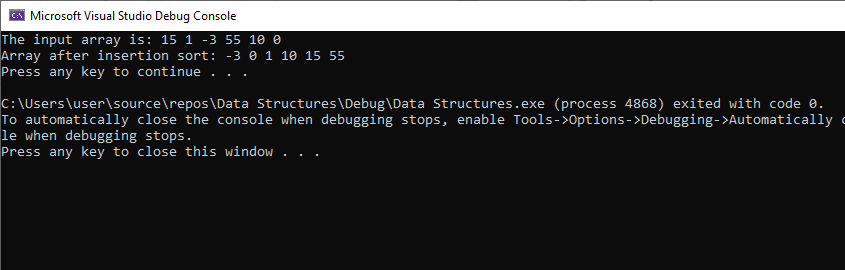
arr[i] = index;

if (first + 1 <= size) {

insertsort(arr, first + 1, size);

}

}



**Task # 04**

#include<iostream>

using namespace std;

int fibonacci(int n)

{

if (n == 0 || n == 1)

{

return n;

}

else

{

return (fibonacci(n - 1) + fibonacci(n - 2));

}

}

int main()

{

int n, i = 0;

cout << "Enter Size of fibonacci series: ";

cin >> n;

cout << "The fibonacci series is: ";

while (i < n)

{

cout << fibonacci(i) << " ";

i++;

}

cout << endl;

}

