|  |
| --- |
|  |
| Assignment 1 |
| **Roll no : f17-8195** |
| **usama rao**  **Section A** |
| **9/28/2018** |

**Task 1:**

1. **(N+2)\*(N-1+2)==N\*N**

**We drop the primitive operation because they are in accurate and inefficient.**

1. **(N+3)\*(N^2+3)\*(N^2+4)== N\*N\*N\*N\*N.**

**Every primitive object cost is different. So, primitive operation is inefficient. So, we drop the significant.**

**Iii) (N+2)\*(N+5)==N\*N**

**Iv) (N)\*(N+4)==N\*N**

**v) (N+3)=N**

**Task 2:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int arr1[2][2]={1,2,3,4};

int arr2[2][2]={1,2,3,4};

int sum[2][2]={0};

for(int i=0; i<2; i++)

{

for(int j=0; j<2; j++)

{

sum[i][j]=arr1[i][j]+arr2[i][j];

cout<<sum[i][j]<<" ";

}

cout<<endl;

}

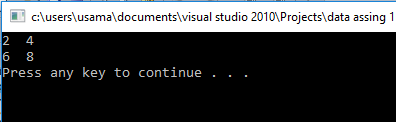
system("pause");

return 0;

}

**Answer: it cost is O(1).**

**Result:**



**Task 3:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int sum=0,line=0;

int i=0,j=0,k=0;

int a[2][2]={{1,2},{3,4}};

int b[2][2]={{5,6},{7,8}};

int c[2][2]={0};

for(i=0; i<2; i++)

{

for(j=0; j<2; j++)

{

for(k=0; k<2; k++)

{

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

for(int i=0; i<2; i++)

{

for(int j=0; j<2; j++)

{

cout<<c[i][j]<<" ";

}

cout<<endl;

}

system("pause");

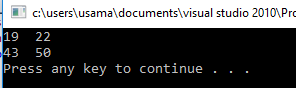
return 0;

}

**Answer:**

**O(1) are the run time of 2d multiplication. Are primitive operation cost inaccurate. So we drop it.**

**Result:**



**Task 4:**

1. **O(n). We reduce primitive operation.**
2. **O(n\*n). We reduce significant bit or primitive operation.**
3. **O(n\*n\*n). We reduce significant and primitive operation cost.**

**Task 5:**

#include<iostream>

#include<conio.h>

using namespace std;

struct node

{

int d;

node \*next;

};

class list

{

public:

node \*head;

node \*tail;

list()

{

head=NULL;

tail=NULL;

}

void insert\_node(int i)

{ //insert node at end

node \*n=new node;

n->d=i;

n->next=NULL;

if(head==NULL)

{

head=n;

tail=n;

}

else

{

tail->next=n;

tail=n;

}

}

void print()

{ //simple print full link list

node \*p=head;

if(head!=NULL)

{

while(p!=NULL)

{

if(p==NULL)

{

return;

}

else

{

cout<<p->d<<endl;

p=p->next;

}

}

}

else

{

cout<<"your list empty, please insert data"<<endl;

}

}

void recursive\_print(node \*p)

{ //recursively print full link list

if(p==NULL)

{

return;

}

else

{

recursive\_print(p->next);

cout<<p->d<<endl;

}

}

void At\_start(int i)

{ //addition of node at start.

node \*n=new node;

n->d=i;

n->next=NULL;

if(head==NULL)

{

head=n;

}

else

{

n->next=head;

head=n;

}

cout<<"insert at strat : ";

}

void At\_mid(int dat)

{ //simple addition of node at mid,stat or at end.

node \*p=head;

if(head==NULL)

{

return;

}

if(dat==0)

{

node \*n=new node;

n->d=dat;

n->next=head;

head=n;

return;

}

else if(head->d==dat)

{

node \*n=new node;

n->d=dat;

n->next=head;

head=n;

return;

}

while(p!=NULL)

{

if(p->d==(dat-1))

{

node \*n=new node;

n->d=dat;

n->next=NULL;

n->next=p->next;

p->next=n;

return;

}

p=p->next;

}

}

void recursive\_at\_mid(int dat,node \*p)

{ //recersive addition of node at mid or at end

node \*n=new node;

n->d=dat;

n->next=NULL;

if(p->next==NULL)

{

return;

}

else if(p->d==(dat-1))

{

n->next=p->next;

p->next=n;

return;

}

p=p->next;

recursive\_at\_mid(dat,p);

}

void deleate\_node(int n)

{

node \*c=head;

node \*p=NULL;

while(c!=NULL && c->d!=n)

{

p=c;

c=c->next;

}

if(c!=NULL)

{

if(p!=NULL)

{

p->next=c->next;

delete c;

}

else

{

head=c->next;

delete c;

}

}

}

bool search(node \*p,int k)

{ //iteratively searching of key.

while(p!=NULL)

{

if(p==NULL)

{

return false;

}

else if(p->d==k)

{

return true;

}

else

{

p=p->next;

}

}

}

bool recursive\_search(node \*p,int k)

{ //recursively searching of key.

if(p==NULL)

{

return false;

}

else if(p->d==k)

{

return true;

}

else

{

recursive\_search(p->next,k);

}

}

void recursive\_search()

{

int k;

cout<<"enetr key that you want to search : ";

cin>>k;

if(recursive\_search(head,k)==1)

{

cout<<"your key is found : "<<endl;

}

else

{

cout<<"your key not found : "<<endl;

}

}

void iterative\_search()

{

int k;

cout<<"enetr key that you want to search : ";

cin>>k;

if(search(head,k)==1)

{

cout<<"your key found"<<endl;

}

else

{

cout<<"your key not found : "<<endl;

}

}

bool recursive\_insertion(node \*p,int n)

{

cin>>n;

if(n==-1)

{

return 1;

}

else

{

p->d=n;

recursive\_insertion(p->next,n);

}

}

};

int main()

{

int n=0;

char a=NULL;

a='a';

while(a=='a'||a=='A')

{

cout<<"0 : for terminate programe"<<endl;

cout<<"1 : for inserting node at end "<<endl;

cout<<"2 : for inserting node at start "<<endl;

cout<<"3 : for inserting node at start, mid or at end"<<endl;

cout<<"4 : for inserting node recursively at mid or at end"<<endl;

cout<<"5 : for searching recursively "<<endl;

cout<<"6 : for searching iteratively "<<endl;

cout<<"7 : for print full link list recursively "<<endl;

cout<<"8 : for print full link list iteratively "<<endl;

cout<<"9 : for recursive insertion "<<endl;

cout<<"10: for delete a particular node"<<endl;

cin>>n;

list l;

system("cls");

if(n==0)

{

break;

}

if(n==1)

{

int k=0;

cout<<"-1 : for return again on previous manu : "<<endl;

cout<<"enter no that you want add at end "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.insert\_node(k);

}

}while(k!=-1);

continue;

}

if(n==2)

{

int k=0;

cout<<"-1 : for return again on previous manu : "<<endl;

cout<<"enter no that you want add at start "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.At\_start(k);

}

}while(k!=-1);

continue;

}

if(n==3)

{

int k=0;

cout<<"enter no that you want add at start or at mid or end with respect to its position : ";

cout<<"-1 : for return again on previous manu : "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.At\_mid(k);

}

}while(k!=-1);

continue;

}

if(n==4)

{

int k=0;

cout<<"enter no that you want add at start or at mid or end with respect to its position : ";

cout<<"-1 : for return again on previous manu : "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.recursive\_at\_mid(k,l.head);

}

}while(k!=-1);

continue;

}

if(n==5)

{

int k=0;

cout<<"enter no that you want to search : ";

cout<<"-1 : for return again on previous manu : "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.recursive\_search();

}

}while(k!=-1);

continue;

}

if(n==6)

{

int k=0;

cout<<"enter no that you want to search : ";

cout<<"-1 : for return again on previous manu : "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.iterative\_search();

}

}while(k!=-1);

continue;

}

if(n==7)

{

int k=0;

l.recursive\_print(l.head);

cout<<"-1 : for return again on previous manu : "<<endl;

cout<<"1 : for print again recursively "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.recursive\_print(l.head);

}

}while(k!=-1);

continue;

}

if(n==8)

{

l.print();

int k=0;

cout<<"-1 : for return again on previous manu : "<<endl;

cout<<"1 : for print again iteratively"<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.print();

}

}while(k!=-1);

continue;

}

if(n==9)

{

int k=0;

cout<<"enter no that you want to insert recursively : ";

cout<<"-1 : for return again on previous manu : "<<endl;

if(l.recursive\_insertion(l.head,k)==1);

{

continue;

}

}

if(n==10)

{

int k=0;

cout<<"enter no that you want to delete : ";

cout<<"-1 : for return again on previous manu : "<<endl;

do

{

cin>>k;

if(k==-1)

{

break;

}

else

{

l.deleate\_node(k);

}

}while(k!=-1);

continue;

}

}

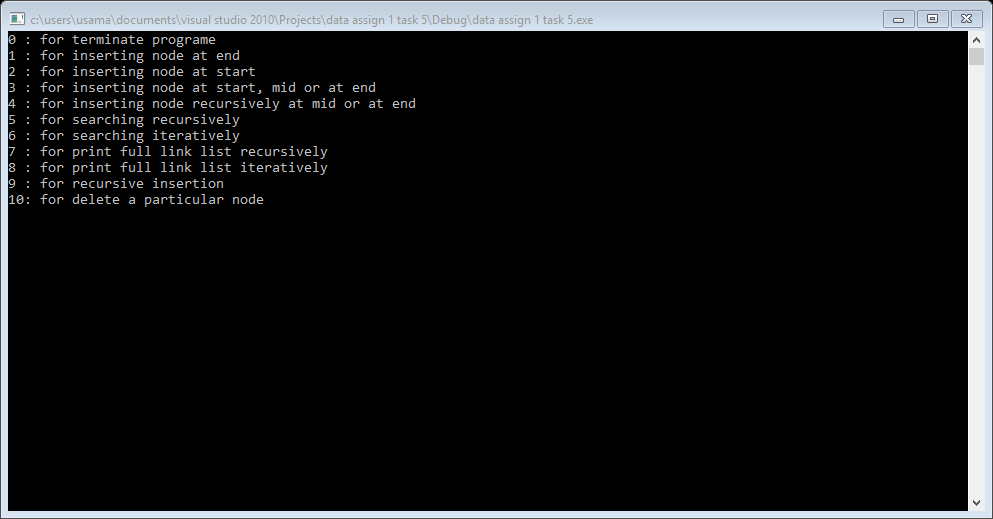
system("pause");

return 0;

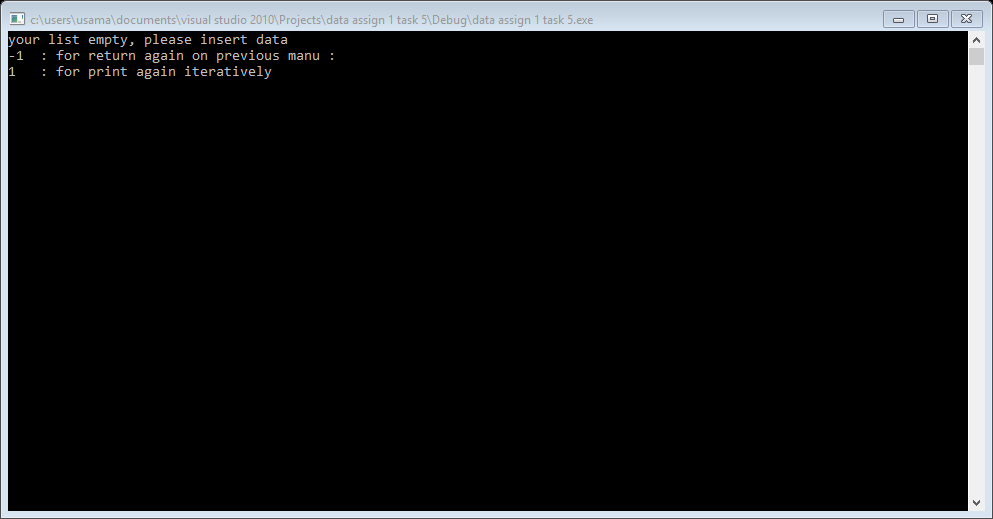
}

**Result:**

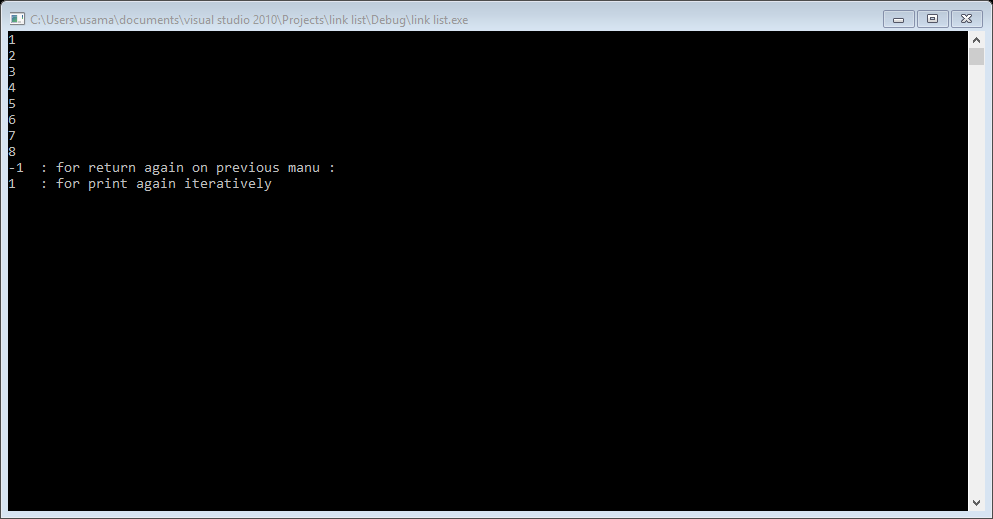
1. **Main manu:**

****

1. **List print with proper message.**

****

1. **Insertion at start, at end at mid etc.**

****

**Task 7:**

#include<iostream>

using namespace std;

struct Node

{

int num;

Node\* next = NULL;

};

Node\* insert\_Node(Node\* list, int x);

void sort\_list(Node\* temp, int size);

void Display(Node\* temp);

Node\* insert\_Node(Node\* head, int num)

{

Node \*NEW = new Node();

NEW->num = num;

NEW->next = NULL;

if (head == NULL)

{

head = NEW;

return head;

}

Node \*pointer = head;

while (pointer->next != NULL)

{

pointer = pointer->next;

}

pointer->next = NEW;

return head;

}

void Display(Node\* head)

{

Node\* temp = new Node();

temp = head;

while (temp != NULL)

{

cout << temp->num << " ";

temp = temp->next;

}

cout << endl;

}

void sort\_list(Node\* head, int size)

{

Node\* temp = new Node();

int carry;

Node\* temp1 = new Node();

temp = head;

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

{

temp = head;

while (temp->next != NULL)

{

temp1 = temp->next;

if (temp->num > temp1->num)

{

carry = temp->num;

temp->num = temp1->num;

temp1->num = carry;

}

temp = temp->next;

}

}

}

int main()

{

Node\* List1 = NULL;

Node\* list2 = NULL;

int input, size1, size2;

cout << "Enter the size of list 1: ";

cin >> size1;

for (int i = 0; i < size1; i++)

{

cout << "///Input Element of list 1:" << endl << i + 1 << " :";

cin >> input;

List1 = insert\_Node(List1, input);

}

cout << "\n\nEnter the size of list 2: ";

cin >> size2;

for (int i = 0; i < size2; i++)

{

cout << "Input Element of list 2:" << endl << i + 1 << " :";

cin >> input;

list2 = insert\_Node(list2, input);

}

sort\_list(List1, size1);

sort\_list(list2, size2);

cout << "\n\n ////SORTED LIST////\n";

cout << "List 1:\n";

Display(List1);

cout << "List 2:\n";

Display(list2);

Node \*merge = List1;

while (merge->next)

{

merge = merge->next;

}

merge->next = list2;

size1 = size1 + size2;

sort\_list(List1, size1);

cout << "\n\n //// MERGED ARRAY \n";

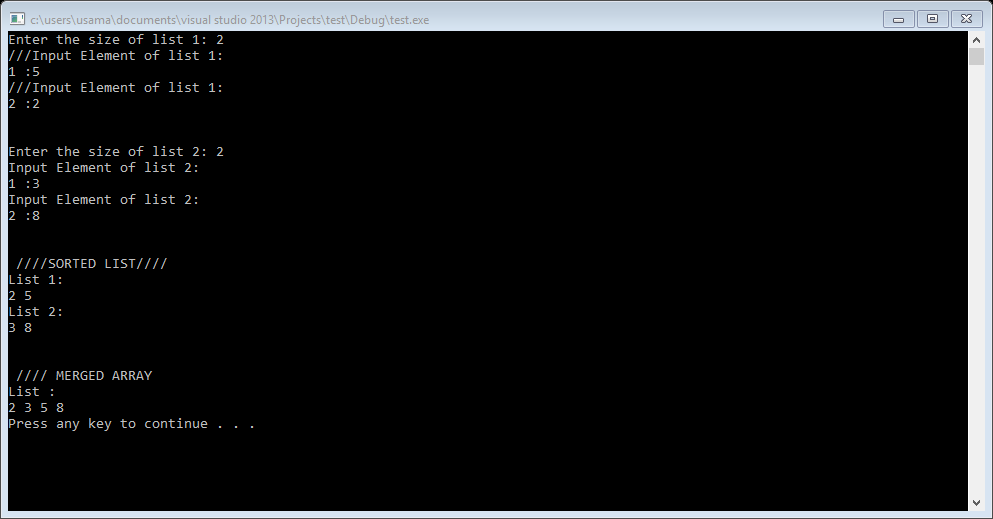
cout << "List :\n";

Display(List1);

system("pause");

}

**Result:**

****

**Task 8:**

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node \*next;

};

Node \*newNode(int data)

{

Node \*temp = new Node;

temp->next = temp;

temp->data = data;

return temp;

}

void position(int m, int n)

{

Node \*head = newNode(1);

Node \*prev = head;

for (int i = 2; i <= n; i++)

{

prev->next = newNode(i);

prev = prev->next;

}

prev->next = head;

Node \*ptr1 = head, \*ptr2 = head;

while (ptr1->next != ptr1)

{

int count = 1;

while (count != m)

{

ptr2 = ptr1;

ptr1 = ptr1->next;

count++;

}

ptr2->next = ptr1->next;

ptr1 = ptr2->next;

}

cout << "Last person is: " << ptr1->data;

cout << endl;

}

int main()

{

int n = 4, m = 2;

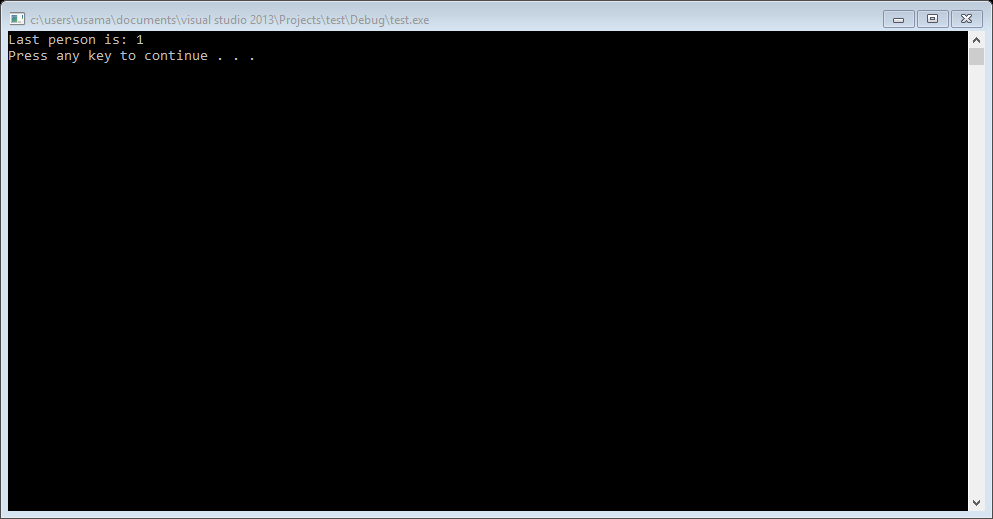
position(m, n);

system("pause");

return 0;

}

**Result:**

****

**Task 11:**

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node \*next;

};

Node \*newNode(int data)

{

Node \*temp = new Node;

temp->next = temp;

temp->data = data;

return temp;

}

void position(int m, int n)

{

Node \*head = newNode(1);

Node \*prev = head;

for (int i = 2; i <= n; i++)

{

prev->next = newNode(i);

prev = prev->next;

}

prev->next = head;

Node \*ptr1 = head, \*ptr2 = head;

while (ptr1->next != ptr1)

{

int count = 1;

while (count != m)

{

ptr2 = ptr1;

ptr1 = ptr1->next;

count++;

}

ptr2->next = ptr1->next;

ptr1 = ptr2->next;

}

cout << "Last person is: " << ptr1->data;

cout << endl;

}

int main()

{

int n = 4, m = 2;

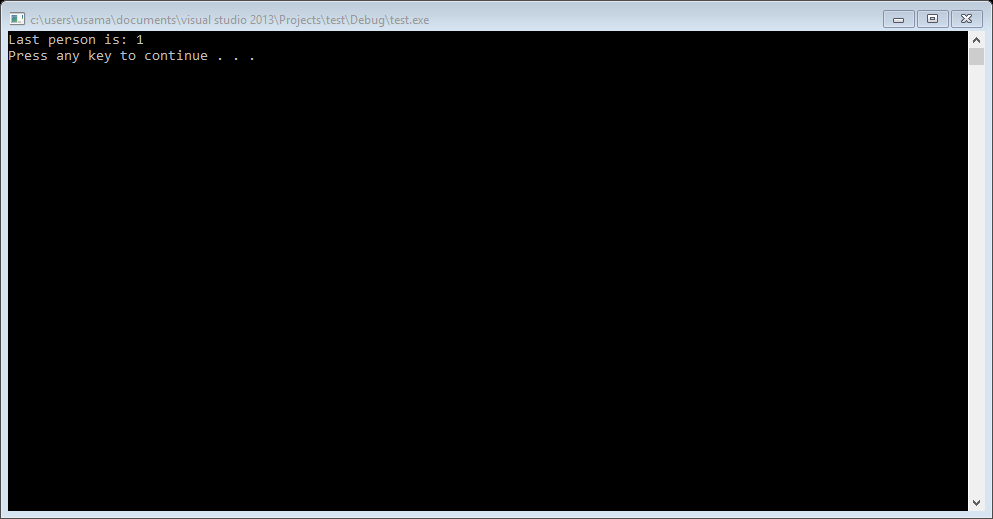
position(m, n);

system("pause");

return 0;

}

**Result:**

****