# Question no 1

#include<iostream>

using namespace std;

int main() {

int a[10];

int data=0;

cout << "Enter the Data is ";

cout << endl;

int count = 0;

int count1 = 0;

for (int i = 0; i < 10; i++) {

cin >> data;

a[i] = data;

if (i == 9) {

break;

}

if (data % 2 == 0)

{

count = count + 1;

}

if (data % 2 != 0)

{

count1 = count1 + 1;

}

if (count == 5) {

cout << "All Evene Indexes are occupied";

cout << endl;

cout << "Enter The Odd Value";

cout << endl;

}else if (count1 == 5) {

cout << "All Odd Indexes are occupied";

cout << endl;

cout << "Enter The Even Value";

cout << endl;

}

}

int b[10];

for (int i = 0; i < 10; i++) {

b[i] = 0;

}

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

if (b[j] == 0) {

if (j % 2 == 0) {

if (a[i] % 2 == 0) {

b[j] = a[i];

break;

}

}

}

if (b[j] == 0) {

if (j % 2 != 0) {

if (a[i] % 2 != 0) {

b[j] = a[i];

break;

}

}

}

}

}

cout << "Your Data is ";

cout << endl;

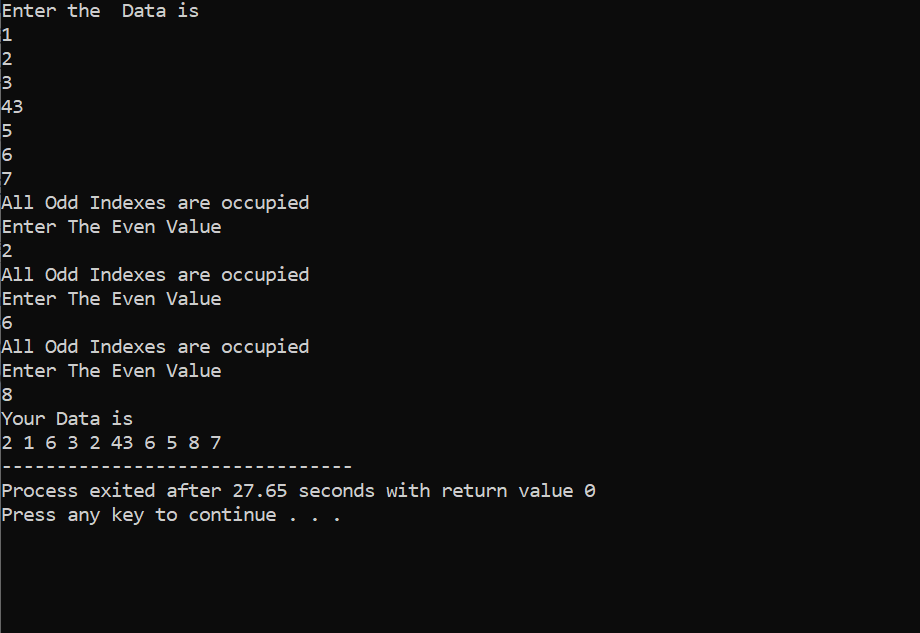
for (int i = 0; i < 10; i++) {

cout << b[i] << " ";

}

};

# OUTPUT



# Question no 2

#include<iostream>

using namespace std;

int main() {

int\* a;

cout << "Enter the Size of the Array";

cout << endl;

int size;

cin >> size;

a = new int[size];

cout << "Enter The Data";

cout << endl;

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

cin >> a[i];

}

int c = 0;

for (int i = 1; i < 10; i++) {

if (a[c] % i == 0) {

cout << "Factor of ::" << a[c] << " ::" << i;

cout << endl;

}

if (i == 9) {

i = 0;

c = c + 1;

}

if (c == size-1) {

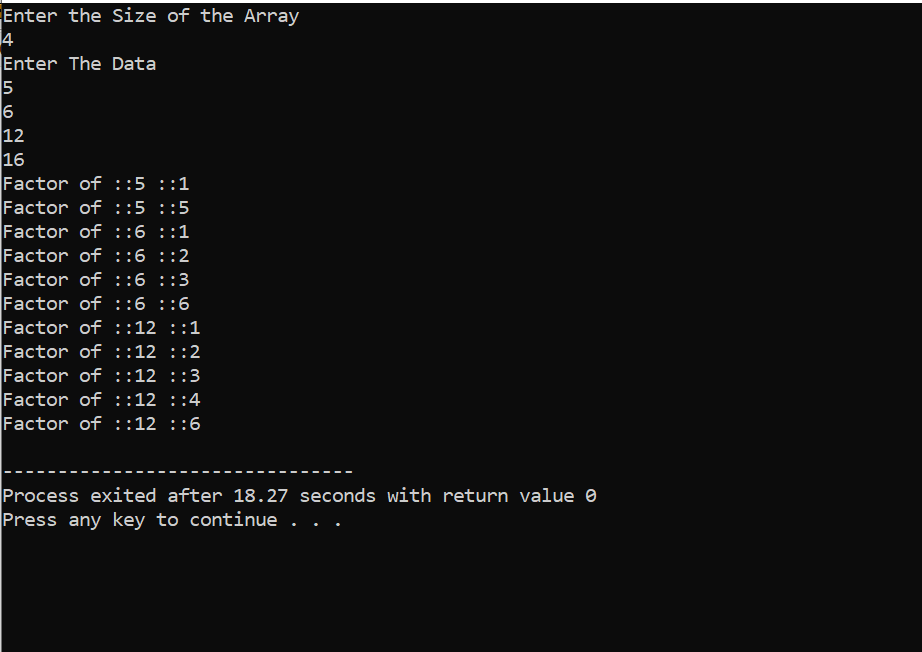
break;

}

}

}

# OUTPUT



# QUESTION NO 3

#include<iostream>

using namespace std;

struct node {

int data;

node\* next;

};

node\* head = NULL;

node\* head1 = NULL;

node\* curr1=NULL;

node\* Insert(node\* NODE, int data) {

node\* curr = NULL;

if (head == NULL) {

NODE = new node;

NODE->data = data;

head = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

curr1 = NODE->next;

}

return NODE;

}

node\* Insert1(node\* NODE, int data) {

node\* curr = NULL;

if (head1 == NULL) {

NODE = new node;

NODE->data = data;

head1 = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

}

return NODE;

}

node\* merge\_lists(node\* NODE1) {

if (NODE1->next== NULL) {

NODE1->next = head1;

}

curr1 = NODE1->next;

return head;

}

void Sort() {

node\* temp = head;

node\* temp2 = head;

temp2 = temp2->next;

node\* Swap = NULL;

node\* check;

while (temp != NULL) {

while (temp2 != NULL) {

if (temp->data < temp2->data) {

swap(temp->data, temp2->data);

}

else {

temp2 = temp2->next;

}

}

temp = temp->next;

temp2 = temp;

if (temp2 == NULL) {

break;

}

temp2 = temp2->next;

}

}

void Print() {

node\* NODE = head;

cout << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

void Print1() {

node\* NODE = head;

cout << endl;

while (NODE != curr1) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

void Print3() {

node\* NODE = head1;

cout << "Your Data is ";

cout << endl;

cout << "---------------";

cout << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

int main() {

node\* Node = NULL;

node\* Node1 = NULL;

int l = 10;

while (true) {

cout << "Enter 1 to Input the Value in 1st Link list";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 2 to Input the Value in 1st Link list";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 3 to Merg and sort";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 4 to Print The Merig Link List I.E merg should be done";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 5 to Print The First Link List";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 6 to Print The Second Link List";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

int data;

cin >> data;

if (data == 1) {

cout << "Enter The Data You Want to Insert";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

int data=0;

cin >> data;

Node = Insert(Node, data);

}

if (data == 2) {

cout << "Enter The Data You Want to Insert";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

int data1=0;

cin >> data1;

Node1 = Insert1(Node1, data1);

}

if (data == 3) {

merge\_lists(Node);

Sort();

}

if (data == 4) {

Print();

}

if (data == 5) {

Print1();

}

if (data == 6) {

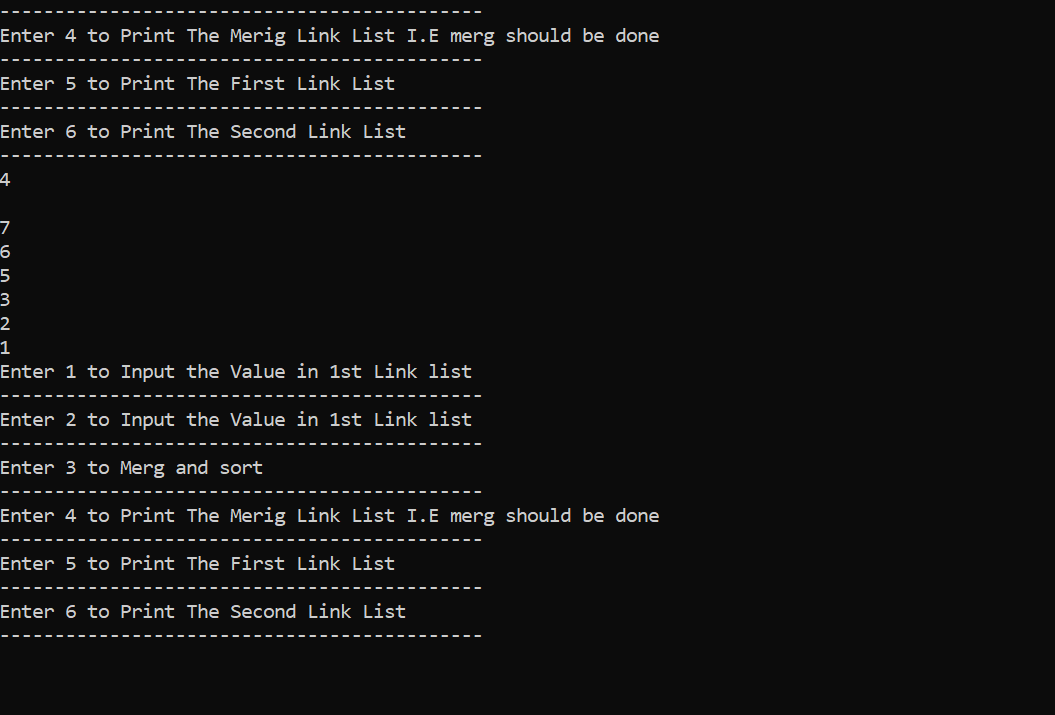
Print3();

}

}

}

# OUTPUT



# QUESTION NO 4

#include<iostream>

using namespace std;

struct node {

int data;

node\* next;

};

node\* head = NULL;

node\* head1 = NULL;

node\* curr1 = NULL;

node\* head2 = NULL;

node\* Insert(node\* NODE, int data) {

node\* curr = NULL;

if (head == NULL) {

NODE = new node;

NODE->data = data;

head = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

curr1 = NODE;

}

return NODE;

}

node\* Insert1(node\* NODE, int data) {

node\* curr = NULL;

if (head1 == NULL) {

NODE = new node;

NODE->data = data;

head1 = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

}

return NODE;

}

node\* Insert3(node\* NODE, int data) {

node\* curr = NULL;

if (head2 == NULL) {

NODE = new node;

NODE->data = data;

head2 = NODE;

curr = NODE;

}

else {

curr = NODE;

NODE = new node;

NODE->data = data;

NODE->next = NULL;

curr->next = NODE;

}

return NODE;

}

node\* merge\_lists(node\* NODE1) {

node\* temp = head1;

int count = 0;

while (count < 3) {

if (count == 2) {

curr1->next = temp;

break;

}

else {

temp = temp->next;

count = count + 1;

}

}

return head;

}

void Print() {

node\* NODE = head;

cout << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

void Common\_Element() {

head2 = NULL;

node\* NODE = head;

node\* NODE2 = NULL;

while (NODE != NULL) {

NODE2 = Insert3(NODE2, NODE->data);

NODE = NODE->next;

}

node\* temp2 = head2;

node\* temp3 = head2;

int count = 0;

int data = 0;

while (temp2 != NULL) {

temp3 = head2;

data = temp2->data;

while (temp3 != NULL) {

if (temp3->data == data) {

count = count + 1;

temp3->data = 0;

}

temp3 = temp3->next;

}

if (data != 0 && count > 1) {

cout << "Your Data :: " << data << " Comes " << count << " Times";

cout << endl;

}

count = 0;

temp2 = temp2->next;

}

cout << endl;

}

void Print1() {

node\* NODE = head;

cout << endl;

while (NODE != curr1->next) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

void Print3() {

node\* NODE = head1;

cout << "Your Data is ";

cout << endl;

cout << "---------------";

cout << endl;

while (NODE != NULL) {

cout << NODE->data;

cout << endl;

NODE = NODE->next;

}

}

int main() {

node\* Node = NULL;

node\* Node1 = NULL;

int l = 10;

while (true) {

cout << "Enter 1 to Input the Value in 1st Link list";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 2 to Input the Value in 1st Link list";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 3 to Merg ";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 4 to Print The Merig Link List i.e Mergigng should done";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 5 to Print The First Link List";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 6 to Find The Common Element";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 7 to Print The Second Link List";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

cout << "Enter 8 to Break";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

int data;

cin >> data;

if (data == 1) {

cout << "Enter The Data You Want to Insert";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

int data = 0;

cin >> data;

Node = Insert(Node, data);

}

if (data == 2) {

cout << "Enter The Data You Want to Insert";

cout << endl;

cout << "--------------------------------------------";

cout << endl;

int data1 = 0;

cin >> data1;

Node1 = Insert1(Node1, data1);

}

if (data == 3) {

merge\_lists(Node1);

}

if (data == 4) {

Print();

}

if (data == 5) {

Print1();

}

if (data == 6) {

Common\_Element();

}

if (data == 7) {

Print3();

}

if (data == 8) {

break;

}

}

}

# OUTPUT

Text

Description automatically generated

Text

Description automatically generated

# QUESTION NO 5

#include<iostream>

using namespace std;

struct node

{

int data;

struct node\* next;

struct node\* prev;

}

\*start;

class double\_llist

{

private:

node\* head = NULL;

node\* current\_ptr = NULL;

int count;

public:

node\* insert\_at\_begin(int value, node\* temp) {

if (head == NULL) {

temp = new node;

temp->data = value;

head = temp;

current\_ptr = temp;

temp->next = NULL;

temp->prev = NULL;

}

else {

current\_ptr = temp;

temp = new node;

temp->data = value;

current\_ptr->next = temp;

temp->prev = current\_ptr;

temp->next = NULL;

}

return temp;

}

void insert\_after(int value, int position) {

node\* Temp = head;

int counter = 0;

node\* temp2 = NULL;

node\* temp1 = new node;

while (Temp != NULL) {

if (position == 0) {

temp1->data = value;

temp1->prev = Temp;

temp2 = Temp;

temp2 = temp2->next;

temp1->next = temp2;

Temp->next = temp1;

break;

}

if (counter == position) {

temp1->prev = Temp;

temp2 = Temp;

temp2 = temp2->next;

temp1->data = value;

temp1->next = temp2;

Temp->next = temp1;

break;

}

Temp = Temp->next;

counter = counter + 1;

}

}

node\* insert\_at\_end(int value, node\* NODE) {

node\* temp = new node;

temp->data = value;

NODE->next = temp;

temp->prev = NODE;

temp->next = NULL;

return temp;

};

void delete\_at\_begin() {

node\* temp = head;

if (temp == NULL) {

delete temp;

return;

}

temp = temp->next;

delete head;

if (temp == NULL) {

head = NULL;

return;

}

head = temp;

temp->prev = NULL;

};

void delete\_before(int value) {

node\* Temp = head;

node\* temp2 = NULL;

node\* temp3 = NULL;

while (Temp != NULL) {

if (Temp->data == value) {

temp3 = Temp;

temp2 = Temp;

temp2 = Temp->prev;

temp3 = Temp->next;

if (temp3 == NULL) {

break;

}

delete Temp;

temp3->prev = temp2;

temp2->next = temp3;

Temp = temp3;

break;

}

Temp = Temp->next;

}

};

int Search(int data) {

node\* Temp = head;

int count = 0;

while (Temp != NULL) {

if (data == Temp->data) {

return count;

}

else {

Temp = Temp->next;

count = count + 1;

}

}

}

node\* delete\_at\_end(node\* NODE) {

node\* temp = NODE;

if (temp == NULL) {

head = NULL;

return head;

}

temp = temp->prev;

delete NODE;

if (temp == NULL) {

head = NULL;

return head;

}

temp->next = NULL;

return temp;

};

void display\_dlist() {

node\* Temp = head;

cout << "Your Data is ";

cout << endl;

cout << "-------------------";

cout << endl;

while (Temp != NULL) {

cout << Temp->data << " ";

Temp = Temp->next;

}

cout << endl;

}

bool is\_empty();

double\_llist()

{

head = NULL;

current\_ptr = NULL; //constructor for my class

count = 0;

}

~double\_llist() {

current\_ptr = head;

while (current\_ptr != NULL)

{

node\* next = current\_ptr->next;

delete current\_ptr;

current\_ptr = next;

}

}

};

int main() {

node\* NODE = NULL;

double\_llist a;

while (true) {

cout << "Enter the 1 To insert The Data\n";

cout << "-------------------------------\n";

cout << "Enter the 2 To insert The Data at position \n";

cout << "-------------------------------------------\n";

cout << "Enter the 3 To insert The Data at last position \n";

cout << "-----------------------------------------------\n";

cout << "Enter the 4 To Delete At the First Position \n";

cout << "-----------------------------------------------\n";

cout << "Enter 5 to Delete At the Last Position \n";

cout << "-----------------------------------------------\n";

cout << "Enter 6 Delete At Value Except Last\n";

cout << "-----------------------------------------------\n";

cout << "Enter 7 to Display At the The List\n";

cout << "-----------------------------------------------\n";

cout << "Enter 8 to Search\n";

cout << "-----------------------------------------------\n";

cout << "Enter 9 to Break\n";

cout << "-----------------------------------------------\n";

int in;

cin >> in;

if (in == 1) {

cout << "Enter the Data you want to Insert\n";

cout << "-----------------------------------------------\n";

int data1;

cin >> data1;

NODE = a.insert\_at\_begin(data1, NODE);

}

if (in == 2) {

cout << "Enter the Data At the n Position\n";

cout << "-----------------------------------------------\n";

cout << "Enter the Position\n";

cout << "-------------------------------------------------\n";

int pos;

cin >> pos;

cout << "Enter The Data To Insert At the Last position\n";

cout << "-------------------------------------------------\n";

int data;

cin >> data;

a.insert\_after(data, pos);

}

if (in == 3) {

cout << "Enter the Data you want to Insert at the Last positon\n";

cout << "-----------------------------------------------\n";

int data;

cin >> data;

NODE = a.insert\_at\_end(data, NODE);

}

if (in == 4) {

a.delete\_at\_begin();

}

if (in == 5) {

NODE = a.delete\_at\_end(NODE);

}

if (in == 6) {

cout << "Enter value You Want to Delete\n";

cout << "-----------------------------------------------\n";

int pos;

cin >> pos;

a.delete\_before(pos);

}

if (in == 7) {

a.display\_dlist();

}

if (in == 8) {

cout << "Enter value You Want to Search\n";

cout << "-----------------------------------------------\n";

int data;

cin >> data;

cout << "Your Data is at position \n";

cout << "-----------------------------------------------\n";

cout << a.Search(data);

cout << endl;

}

if (in == 9) {

break;

}

}

system("pause");

return 0;

}

# OUTPUT

Text

Description automatically generated

# QUESTION NO 6

#include<iostream>

#include<cstring>

#include<string>

using namespace std;

struct node

{

char data;

struct node\* next;

struct node\* prev;

}

\*start;

class double\_llist

{

private:

node\* head = NULL;

node\* current\_ptr = NULL;

node\* head1 = NULL;

node\* current\_ptr1 = NULL;

int count;

public:

node\* insert\_at\_begin(int value, node\* temp) {

if (head == NULL) {

temp = new node;

temp->data = value;

head = temp;

current\_ptr = temp;

temp->next = NULL;

temp->prev = NULL;

}

else {

current\_ptr = temp;

temp = new node;

temp->data = value;

current\_ptr->next = temp;

temp->prev = current\_ptr;

temp->next = head;

current\_ptr1 = temp;

}

return temp;

}

node\* delete\_at\_begin() {

node\* temp = head;

temp = temp->next;

delete head;

head = temp;

current\_ptr1->next = head;

return temp;

};

void Vowel\_Sorting() {

string oval = {};

node\* Temp = head;

char data = Temp->data;

int i = 0;

while (data != 'A' && data != 'a' && data != 'E' && data != 'e' && data != 'I' && data != 'i' && data != 'O' && data != 'o' && data != 'U' && data && 'u') {

oval = oval + data;

Temp = delete\_at\_begin();

data = Temp->data;

if (Temp == current\_ptr1) {

break;

}

}

if ((data != 'A' && data != 'a' && data != 'E' && data != 'e' && data != 'I' && data != 'i' && data != 'O' && data != 'o' && data != 'U' && data && 'u')||oval == "\0") {

cout << oval;

display\_dlist();

cout << "-";

cout << "way";

}

else if (oval != "\0") {

display\_dlist();

cout << "-";

oval = oval + "ay";

cout << oval;

}

cout << endl;

}

void display\_dlist() {

node\* Temp = head;

cout << Temp->data << " ";

Temp = Temp->next;

while (Temp != head) {

cout << Temp->data << " ";

Temp = Temp->next;

}

}

bool is\_empty();

double\_llist()

{

head = NULL;

current\_ptr = NULL; //constructor for my class

count = 0;

}

~double\_llist() {

current\_ptr = head;

while (current\_ptr != current\_ptr1)

{

node\* next = current\_ptr->next;

delete current\_ptr;

current\_ptr = next;

}

}

};

int main() {

node\* NODE = NULL;

double\_llist a;

string input;

cout << "Input the String ";

cout << endl;

cout << "------------------";

cout << endl;

cin >> input;

for (int i = 0; input[i] != '\0'; i++) {

NODE = a.insert\_at\_begin(input[i], NODE);

}

a.display\_dlist();

cout << endl;

a.Vowel\_Sorting();

system("pause");

return 0;

}

# OUTPUT

Text

Description automatically generated with medium confidence