/\*Given a singly linked list of size **N** containing only English Alphabets, arrange the consonants and vowel nodes of the list it in such a way that all the vowels nodes come before the consonants while maintaining the **order of their arrival**.

**1.     Input:** a e g h i m;          **Output:** a e i g h m

**2.     Input:** q r t;             **Output:** q r t

\*/

#include<iostream.h>

#include<conio.h>

struct Node

{

char data;

struct Node \*next;

};

/\* Function to add new node to the List \*/

Node \*newNode(char key)

{

Node \*temp = new Node;

temp->data = key;

temp->next = NULL;

return temp;

}

// utility function to print linked list

void printlist(Node \*head)

{

if (! head)

{

cout << "Empty List";

return;

}

while (head != NULL)

{

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

if (head->next)

cout << "-> ";

head = head->next;

}

cout << endl;

}

// utility function for checking vowel

int isVowel(char x)

{

return (x == 'a' || x == 'e' || x == 'i' ||

x == 'o' || x == 'u');

}

/\* function to arrange consonants and

vowels nodes \*/

Node \*arrange(Node \*head)

{

Node \*newHead = head;

// for keep track of vowel

Node \*latestVowel;

Node \*curr = head;

if (head == NULL)

return NULL;

// We need to discover the first vowel

if (isVowel(head->data))

latestVowel = head; //if first element is vowel it will be the head

else //if first elemt is not vowel iterate throught the list to find the next vowel

{

// Note that curr points to the element

// \*before\* the element with the vowel.

while (curr->next != NULL && !isVowel(curr->next->data))

curr = curr->next;

if (curr->next == NULL)

return head;

latestVowel = newHead = curr->next;

curr->next = curr->next->next;

latestVowel->next = head;

}

while (curr != NULL && curr->next != NULL)

{

if (isVowel(curr->next->data))

{

// The next discovered item is a vowel

if (curr == latestVowel)

{

latestVowel = curr = curr->next;

}

else

{

Node \*temp = latestVowel->next;

latestVowel->next = curr->next;

latestVowel = latestVowel->next;

curr->next = curr->next->next;

latestVowel->next = temp;

}

}

else

{

curr = curr->next;

}

}

return newHead;

}

void main()

{

clrscr();

Node \*head = newNode('a');

head->next = newNode('b');

head->next->next = newNode('c');

head->next->next->next = newNode('e');

head->next->next->next->next = newNode('d');

head->next->next->next->next->next = newNode('o');

head->next->next->next->next->next->next = newNode('x');

head->next->next->next->next->next->next->next = newNode('i');

cout<<"ARRANGE VOWELS & CONSONANTS IN A SINGLY LINKED LIST"<<endl;

cout<<"\nLinked list before :";

printlist(head);

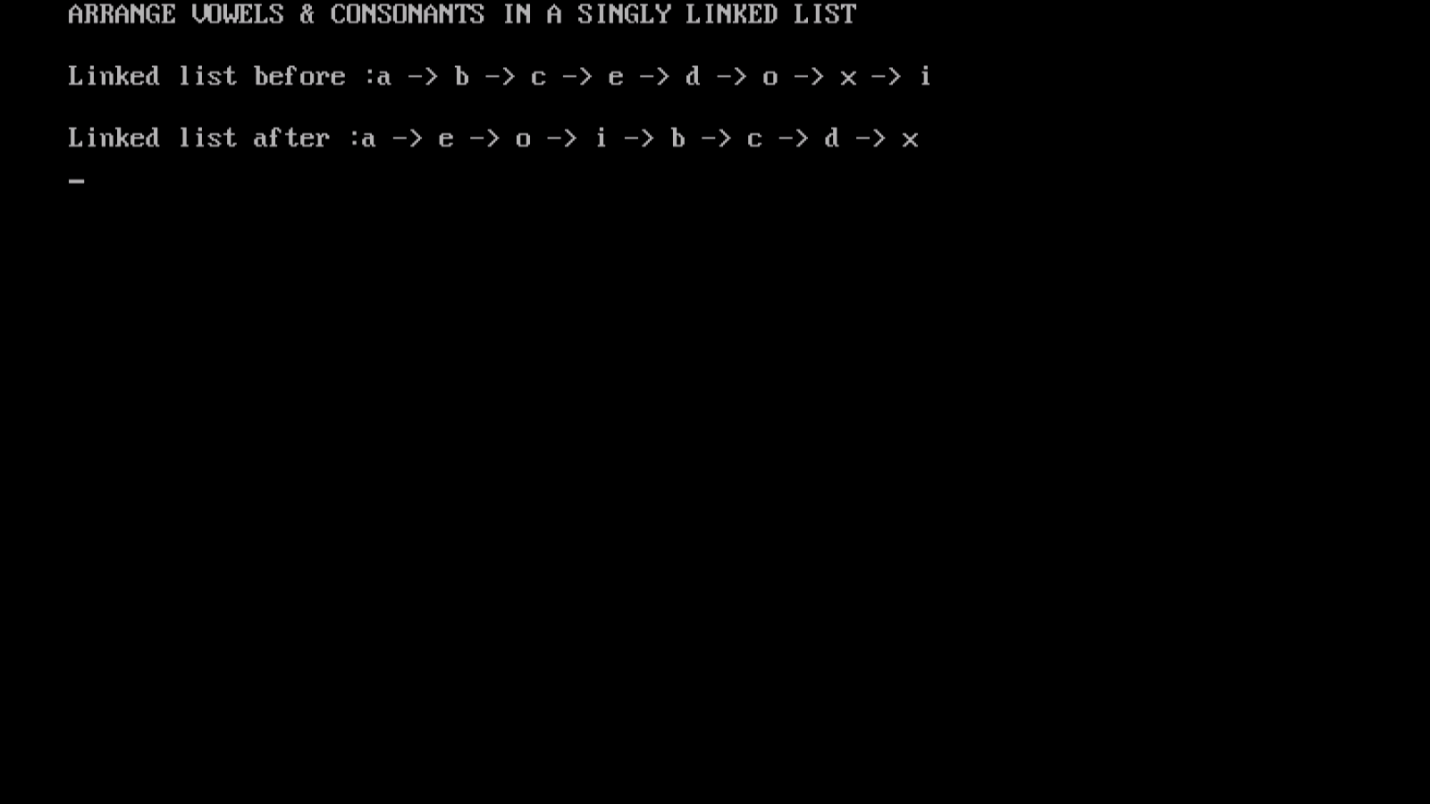
head = arrange(head);

cout<<"\nLinked list after :";

printlist(head);

getch();

}



**//This code was contributed by Vijeyandrian MCA Section ’B’**