George Jr C. Gesite // BSCpE-2A

Problem No 2.

```
# include <sstream>
# include <iostream>
using namespace std;

string str;
int count = 0;

struct Node {
 string data;
  int Key;
 Node* next;
 };

class SLL {
  public:

  Node* head;
  SLL() {
  head = NULL;
  }
  void input() {
   cout << "Input";
   getline (cin, str);
  }
  void Check if Prime () {
   string cut;
   stringstream ss (str);
   while (ss >> cut)
   {   int c= cut.length();
       int count1 = 0, m= 0, flag = 0;
       m= c/2;
       for (int j= 2; j<=m; j++) {
         if ( c % j == 0) {
            flag=1;
       }}
if (flag == 0) {
  InserNode (cut);
}
 }
cout << " word(s) whose length is a prime number: " << endl;
Node* temp = head;
 while (temp != NULL)
 { cout << "(" << temp->data << ", " << temp->key << ")" << "->";
    temp = temp->next;
  }
}
void InserNode (string n)
  Node * newnode = new Node;
  count ++;
  newnode -> key = count;
  newnode -> data = n;
  newnode -> next = NULL;
  Node* temp = head;
  if ( head == NULL) {
     head = newnode;
   }
  else {
   while ( temp->next != NULL) {
     temp = temp->next; }
    temp->next = newnode;
   }
  }
 void Insert Node at Head (string n) {
  Node* newnode = new Node;
  count ++;
  newnode -> data = n;
  newnode -> key = count;
  newnode -> next = head;
   head = new node;

  }
```

```
void InsertNodeAfter ( int n, string m) {
 Node* newnode = new Node;
 newnode ->data = m;
 count++;
 newnode -> key = count;
 Node* temp = head;
 Node* prev;
 while ( temp->next != NULL) {
   if ( temp->key == n) {
        prev = temp;
        break;
   }
   else {
     temp = temp->next;
     }
   }
   if (prev== NULL)
   return;
   else {
   newnode ->next = prev->next;
   prev ->next = newnode;
   }
 }
 Void ReplaceDataNode (int n, string m) {
 Node* temp = head;
 while ( temp->next != NULL) {
    if (temp->key == n) {
         temp->data = m;
           break;
       }
      else {
      temp = temp->next;
      }
    }
 }
```

```
void delete CertainNode (int n) {
    Node * temp = head ;
    Node * prev = NULL;
    if ( temp! = NULL && temp -> Key = = n) {
       head = terp -> next ;
       delete trp ;
     }
    else {
      while ( temp ! = NULL && trmp -> key ! = n) {
          prev = temp;
          temp = temp -> next ;
        }
      if (temp == NULL)
         return;
       else {
         prev -> next = temp -> next ;
        delete temp;
          }
       }
   }
   void Display () {
     if ( head  == NULL)
          return ;
     else {
       Node* temp = head;
       while ( temp ! = NULL) {
       cout << "(" << temp -> data << "," << temp-> Key << ")" << "->";
         temp = terp -> next;
         }
      }
   }
};
```

```
int main () {
  SLL sll;
  sll. input ();
  sll. Check if Prime ();
  char option ;
  string value;
  int val;
  do {
   cout<<endl<< "Menu Options"<< endl;
   cout << "[a] insert at beginning"<<endl;
   cout << "[b] insert at end "<< endl;
   cout << "[c] insert at a given position "<< endl;
   cout << "[d] update node" << endl;
   cout << "[e] delete" << endl ;
   cout << "[f] Display << endl;
   cout << endl << Enter option: "<< endl;
   cin>> option ;
   switch (option)
   {
       case 0: {
           break;
       }
      case 'a': {
       cout <<"Enter string to be incerted at beginning : " endl;
       cin>> value;
       sll. Insert Node at Head (value);
       break;
      }
     case 'b': {
      cout << " Enter string to be inserted at end: " endl;
       cin >> value
      sll . Inser Node ( value);
     break;
     }
case 'c' : {
   cout<<" Enter string to be inserted at a given position : " <<endl;
   cin>>val;
   cin >> value;
   sll. Insert Node After ( val , value);
      break;
   }
case 'd': {
   cout<<" Enter position and string to be updated : "<< endl;
   cin >> val;
   cin >> value;
   sll. Replace Data Node ( val , value);
   break;
  }
case 'e': {
    cout<<" Enter position to be deleted : "<<endl;
    cin >> val;
    sll. deleteCertain Node ( val);
     break;
  }
case 'f': {
     sll. Display ();
       break;
   }
default : {
   cout "Invalid Option";
  }
 }
} while ( option != 0);
 return 0;
}
```