George Jr. C. Gesite BSCpE - 2A Problem No. 7

```
# include <iostream>
# include <cmath>
using namespace std;
int count = 0;
struct Node {
int data;
int key;
Node* Next;
};
class SLL {
  public:
 Node* head;
 SLL() {
  head = NULL;
  }
void InserNode (int n) {
 Node* newnode = new Node;
 count++;
 newnode -> key = count;
 newnode -> data = n;
 nevnode -> next = NULL;
 Node * temp = head;
 if (head == NULL) {
   head = newnode;
  }
 else {
  while ( temp -> next != NULL)
     { temp = temp -> next;
     }
  temp -> next = newnode;
 }
}
```

```
void InsertNodeatHead(int n){
  Node* newnode = new Node;
  count++;
  newnode->data = n;
  newnode->key = count;
  newnode->next = head;
  head = newnode;
void InsertAfter(int n, int 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, int m) {
  Node * temp = head;
  while (temp->nex != NULL) {
  if (temp -> key == n) {
      temp -> data = m;
          break;
    }
   else {
    temp= temp-> next;
     }
}
}
void DeleteCertain Node (int n) {
Node *temp = head;
Node * prev = NULL;
if ( temp != NULL && temp -> key == n) {
  head= temp->next;
  delete temp;
  }
 else {
  while( temp!=NULL && temp-> key != n) {
        prev = temp
        temp = tmp->next;
    }
   if (temp == NULL)
   return;
   else {
     prev-> next = temp -> next;
     delete temp;
    }
  }
}
```

```
void convertToDecimal ( int n) {
    int m;
    Node* temp = head;
   while (temp->next!=NULL){
        if (temp -> key ==n) {
            m = temp-> data;
            break;
        }
       else {
         temp = temp -> next;
       }
    }
    int decimalNumber = 0, i=0, remainder;
    while ( m!=0)
     remainder = m %10;
     m/=10;
     decimal Number += remainder* pow(2,i);
      i++;
    }
     temp-> data = decimalNumber;
  }
  void Display()
  {  if(head == NULL) {
        return;
     else {
       Node *temp = head;
       while( temp!= NULL){
       cout<<"("<< temp->data<<","<< temp-> key<<")"<<"->";
        temp= temp->next;
     }
   }
}
```

```
void Input() {
  int a;
  cout << "Enter binary numbers to be entered in the list, breaks at zero" << endl;
   for(;;) {
     cin >> a;
     if(a == 0) {
        break;
     }
     else {
       InsertNode(a);
     }
   }
  }
};
int main(){
  SLL sll;
  sll.input;
  char option;
  int val;
  int value;
  do {
    cout << "Menu Options:" << endl;
    cout << "[a] insert at beginning" << endl;
    cout << "[b] insert at end" << endl;
    cout << "[c] insert at given position" << endl;
    cout << "[d] update node" << endl;
    cout << "[e] delete" << endl;
    cout << "[f] convert to binary (give its specific position)" << endl;
    cout << "[g] Display" << endl;
    cout << "Enter option:" << endl;
    cin >> option;
    switch(option) {
      case 0: {
        break;
      }
```

```
case 'a': {
    cout << " Enter binary number to be inserted at beginning" << endl;
    cin >> value;
    sll.InsertNodeatHead(value);
    break;
  }
case 'b': {
    cout << " Enter binary number to be inserted at end " << endl;
    cin >> value;
    sll.InserNode(value);
    break;
  }
case 'c': {
    cout << " Enter binary number to be inserted at a given position: " << endl;
    cin >> val;
    cin >> value;
    sll.InsertNodeAfter(val, value);
    break;
  }
case 'd': {
    cout << "Enter position and binary number to be updated " << endl;
    cin >> val;
    cin >> value;
    sll.ReplaceDataNode(val, value);
    break;
  }
case 'e': {
    cout << " Enter position of binary number to be deleted " << endl;
    cin >> val;
    sll.deleteCertainNode(val);
    break;
  }
```

```
case 'f': {
    cout << "Enter position of binary to be converted to decimal : " << endl;
    cin >> val;
    sll.convertToDecimal(val);
    break;
  }
case 'g': {
    sll.Display;
    break;
  }
default: {
   cout<<" Invalid Option " << endl;
  }
 }
} while(option!=0);
return 0;
}
```