Data Structures (UCS301) Assignment

NAME-Suneet Arora **ROLL NO-1024030174**

```cpp
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(NULL) {}
    };

    Node* head = NULL;

    // Insert at beginning
    void insertBeginning(int x) {
        Node* n = new Node(x);
        n->next = head;
        head = n;
    }

    void display() {
        Node* t = head;
        while(t) {
            cout << t->data << " ";
            t = t->next;
        }
        cout << endl;
    }
};

int main() {
    SLL s;

    s.insertBeginning(30)
    ;
    s.insertBeginning(20)
    ;
    s.insertBeginning(10)
    ;

    s.display(); // Output: 10 20 30
}
#include <bits/stdc++.h>
```

```cpp
using namespace std;

class SLL {
public:
    struct Node {
        int data;
        Node* next;
        Node(int x) : data(x), next(NULL) {}
    };

    Node* head = NULL;

    // Insert at end
    void insertEnd(int x) {
        Node* n = new Node(x);

        if(!head) {
            head = n;
            return;
        }

        Node* t = head;
        while(t->next) t = t->next;

        t->next = n;
    }

    void display() {
        Node* t = head;
        while(t) {
            cout << t->data << " ";
            t = t->next;
        }
        cout << endl;
    }
};

int main() {
    SLL s;

    s.insertEnd(10)
    ;
    s.insertEnd(20)
    ;
    s.insertEnd(30)
    ;

    s.display(); // Output: 10 20 30
```

```
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void
    insertEnd(int x) {
Node* n = new Node(x); if(!head) { head
        = n; return; } Node* t = head;
while(t->next) t = t->next; t->next
        = n;
    }

    // Insert after a given value
    void insertAfter(int key, int x) {
        Node* t = head;
while(t && t->data != key) t = t->next;

if(!t) {
cout << "Key not found!\n"; return;
        }

Node* n = new Node(x); n->next
        = t->next;
t->next = n;
    }

    // Insert before a given value
    void insertBefore(int key, int x) {
        if(!head) return;
```

```
        // If key is head
        if(head->data == key) {
            Node* n = new Node(x);
            n->next = head;
            head = n;
            return;
        }

        Node* prev = NULL;
        Node* curr = head;

        while(curr && curr->data != key) {
            prev = curr;
            curr = curr->next;
        }

        if(!curr) {
            cout << "Key not found!\n";
            return;
        }

        Node* n = new Node(x);
        n->next = curr;
        prev->next = n;
    }

    void display() {
        Node* t = head;
        while(t) {
            cout << t->data << " ";
            t = t->next;
        }
        cout << endl;
    }
};

int main() {
    SLL s;

    s.insertEnd(10)
    ;
    s.insertEnd(20)
    ;
    s.insertEnd(40)
    ;

    s.insertAfter(20, 30);   // 20 → 30 → 40
```

```
    s.insertBefore(40, 35); // 30 → 35 → 40

    s.display();
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void
    insertEnd(int x) {
Node* n = new Node(x);
        if(!head) head = n;
        else {
Node* t = head;
while(t->next) t = t->next; t->next =
            n;
        }
    }

    void deleteBeginning() {
        if(!head) return;

Node* temp = head; head =
        head->next; delete
        temp;
    }

    void display() {
        Node* t = head;
        while(t) {
cout << t->data << " "; t =
            t->next;
        }
cout << endl;
```

```
    }
};

int main() {
SLL s;
    s.insertEnd(10);
    s.insertEnd(20);
    s.insertEnd(30);

    s.deleteBeginning(); // delete 10

    s.display(); // 20 30
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void

    insertEnd(int x) {

        Node* n = new Node(x);
if(!head) head = n; else {
Node* t = head;
while(t->next) t = t->next; t->next =
            n;
        }
    }

    void deleteBeginning() {
        if(!head) return;

Node* temp = head; head =
        head->next;
delete temp;
```

```
    }

    void display() {
        Node* t = head;
        while(t) {
cout << t->data << " "; t =
            t->next;
        }
cout << endl;
    }
};

int main() {
SLL s;
    s.insertEnd(10);
    s.insertEnd(20);
    s.insertEnd(30);

    s.deleteBeginning(); // delete 10

    s.display(); // 20 30
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void
    insertEnd(int x) {
Node* n = new Node(x);
        if(!head) head = n;
        else {
Node* t = head;
```

```
            while(t->next) t = t->next;
            t->next = n;
        }
    }

    void deleteNode(int key) {
        if(!head) return;

        // delete head
        if(head->data == key) {
            Node* temp = head;
            head = head->next;
            delete temp;
            return;
        }

        Node* curr = head;
        Node* prev = NULL;

        while(curr && curr->data != key) {
            prev = curr;
            curr = curr->next;
        }

        if(!curr) {
            cout << "Node not found!\n";
            return;
        }

        prev->next = curr->next;
        delete curr;
    }

    void display() {
        Node* t = head;
        while(t) {
            cout << t->data << " ";
            t = t->next;
        }
        cout << endl;
    }
};

int main() {
    SLL s;
```

```
    s.insertEnd(10);
    s.insertEnd(20);
    s.insertEnd(30);
    s.insertEnd(60);
    s.deleteNode(60);

    s.display(); // 10 20 30

}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void
    insertEnd(int x) {
Node* n = new Node(x);
        if(!head) head = n;
        else {
Node* t = head;
while(t->next) t = t->next; t->next =
            n;
        }
    }

    int search(int key) {
        int pos = 1;
        Node* t = head;

while(t) {
            if(t->data == key)
                return pos;
pos++;
```

```
t = t->next;
        }
return -1;
    }
};

int main() {
SLL s;
    s.insertEnd(10);
    s.insertEnd(20);
    s.insertEnd(30);

    int pos = s.search(20);
if(pos == -1) cout << "Not found\n";
else cout << "Found at position " << pos << endl;
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void

    insertEnd(int x) {

        Node* n = new Node(x);
if(!head) head = n; else {
Node* t = head;
while(t->next) t = t->next; t->next =
            n;
        }
    }

    void display() {
```

```
Node* t = head; while(t) {
cout << t->data << " "; t =
            t->next;
        }
cout << endl;
    }
};

int main() {
SLL s;
    s.insertEnd(10);
    s.insertEnd(20);
    s.insertEnd(30);

    s.display();
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void
    insertEnd(int x) {
Node* n = new Node(x);
        if(!head) head = n;
        else {
Node* t = head;
while(t->next) t = t->next; t->next =
            n;
        }
    }
```

```
// Count occurrences of a key
int countOccurrences(int key) {
    int count = 0;
    Node* t = head;

    while(t) {
        if(t->data == key)
            count++;
        t = t->next;
    }

    return count;
}

// Delete all occurrences of key
void deleteAll(int key) {
    while(head && head->data == key) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }

    Node* curr = head;
    Node* prev = NULL;

    while(curr) {
        if(curr->data == key) {
            prev->next = curr->next;
            delete curr;
            curr = prev->next;
        } else {
            prev = curr;
            curr = curr->next;
        }
    }
}

void display() {
    Node* t = head;
    while(t) {
        cout << t->data << " ";
        t = t->next;
    }
    cout << endl;
}
```

```
};

int main() {
SLL s;

    // Example list
    s.insertEnd(1);
    s.insertEnd(2);
    s.insertEnd(1);
    s.insertEnd(3);
    s.insertEnd(1);
    s.insertEnd(1);

int key = 1;

    int count = s.countOccurrences(key);
cout << "Occurrences of " << key << " = " << count << endl; s.deleteAll(key);

cout << "List after deletion: ";
    s.display();
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
    Node* head = NULL; void
    insertEnd(int x) {
Node* n = new Node(x);
        if(!head) head = n;
        else {
Node* t = head;
while(t->next) t = t->next;
```

```
t->next = n;
        }
    }

    // Find middle using slow-fast pointer
    int findMiddle() {
        if(!head) return -1;

Node* slow = head; Node*
        fast = head;

        while(fast && fast->next) {
            slow = slow->next;
fast = fast->next->next;
        }

return slow->data;
    }
};

int main() {
SLL s;

    s.insertEnd(1);
    s.insertEnd(2);
    s.insertEnd(3);
    s.insertEnd(4);
    s.insertEnd(5);

    cout << "Middle element = " << s.findMiddle();
}
```

```
#include <bits/stdc++.h>
using namespace std;

class SLL {
public:
    struct Node {
int data; Node* next;
        Node(int x) : data(x), next(NULL) {}
    };
```

```
    Node* head = NULL;

    void insertEnd(int x) {
        Node* n = new Node(x);
        if(!head) head = n;
        else {
            Node* t = head;
            while(t->next) t = t->next;
            t->next = n;
        }
    }

    // Find middle using slow-fast pointer
    int findMiddle() {
        if(!head) return -1;

        Node* slow = head;
        Node* fast = head;

        while(fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
        }

        return slow->data;
    }
};

int main() {
    SLL s;

    s.insertEnd(1)
    ;
    s.insertEnd(2)
    ;
    s.insertEnd(3)
    ;
    s.insertEnd(4)
    ;
    s.insertEnd(5)
    ;

    cout << "Middle element = " << s.findMiddle();
}
```