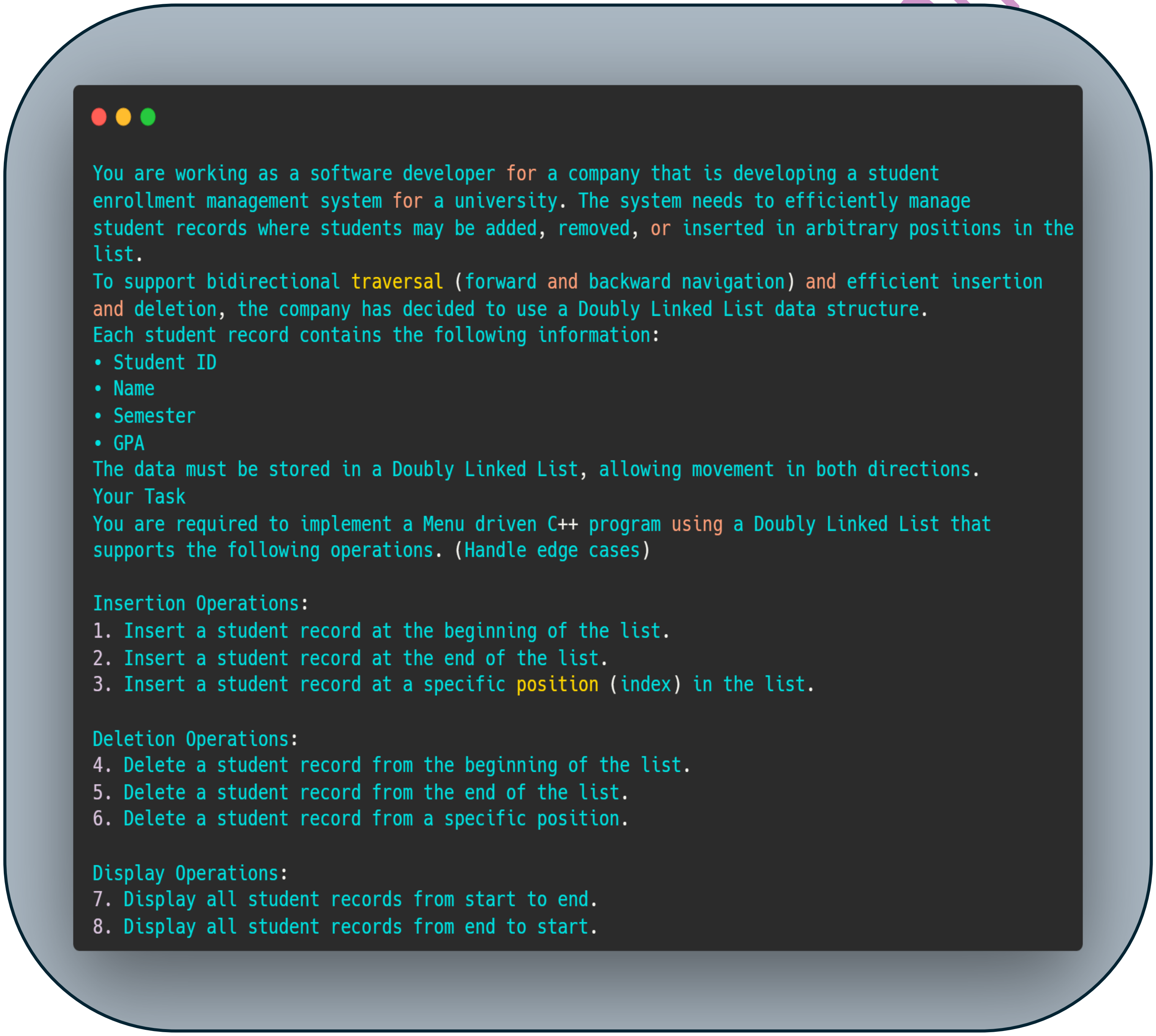
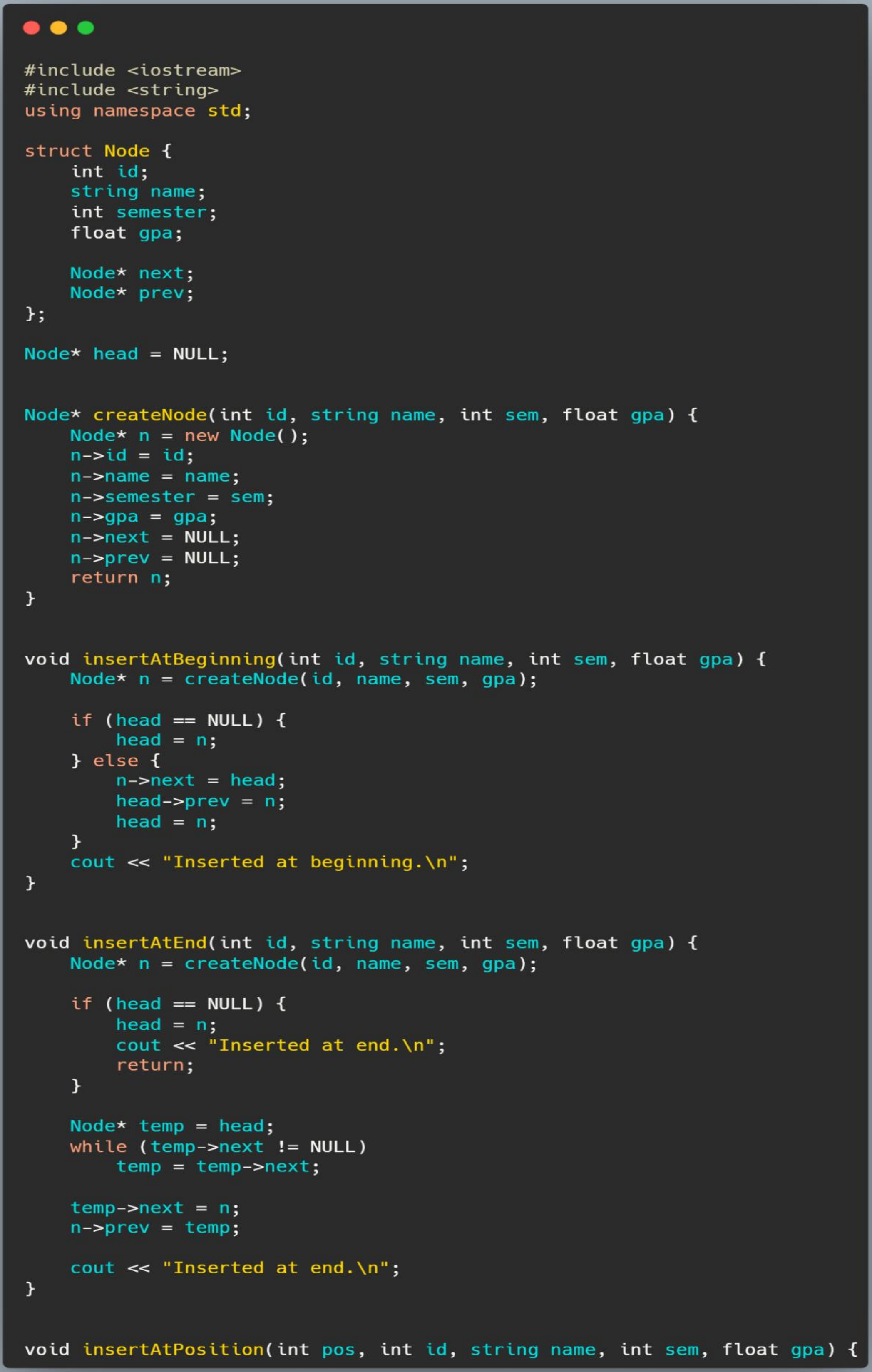
Lab Task 6 [student enrollment management system]

Task:

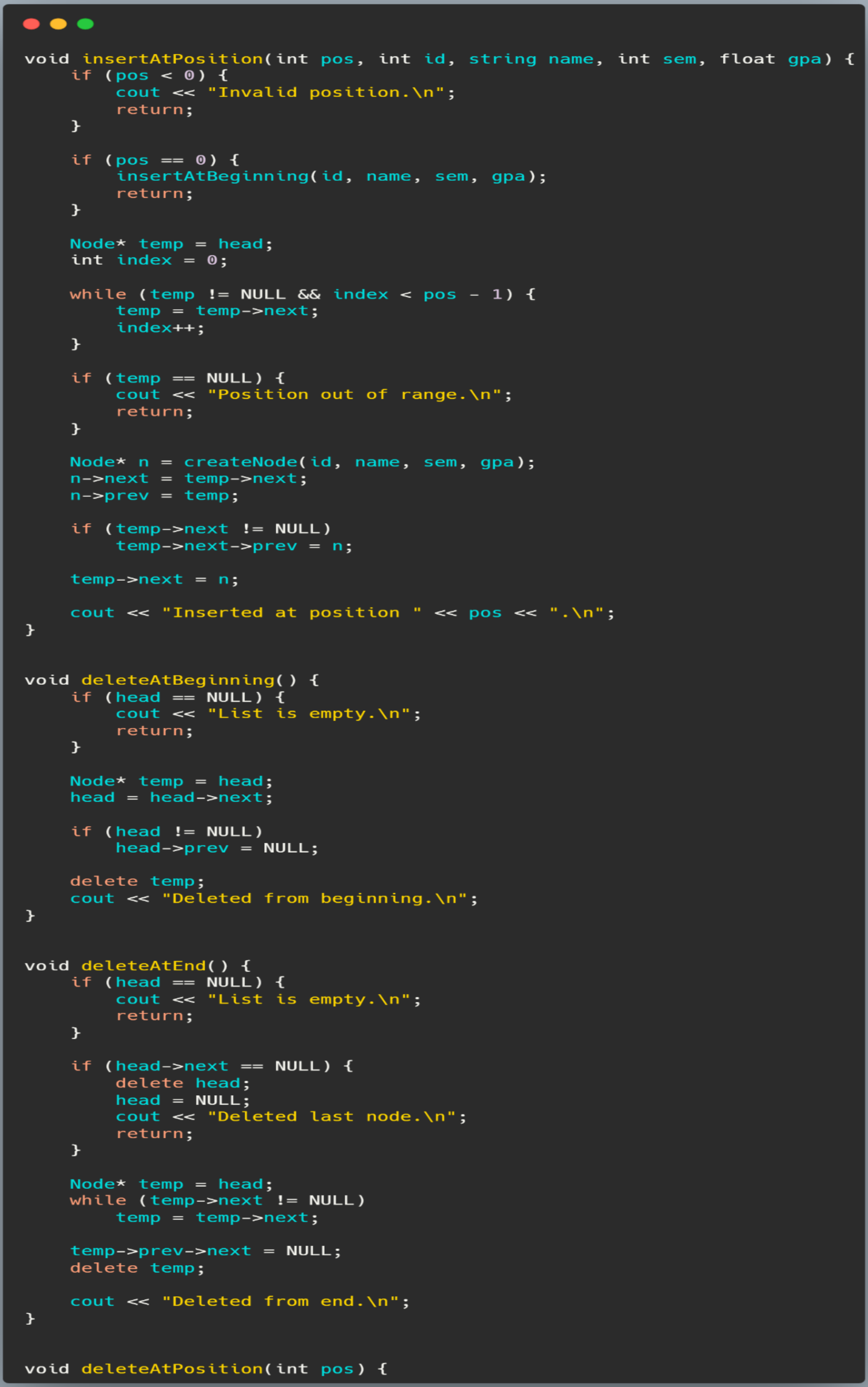


```
You are working as a software developer for a company that is developing a student
enrollment management system for a university. The system needs to efficiently manage
student records where students may be added, removed, or inserted in arbitrary positions in the
list.
To support bidirectional traversal (forward and backward navigation) and efficient insertion
and deletion, the company has decided to use a Doubly Linked List data structure.
Each student record contains the following information:
• Student ID
• Name
• Semester
• GPA
The data must be stored in a Doubly Linked List, allowing movement in both directions.
Your Task
You are required to implement a Menu driven C++ program using a Doubly Linked List that
supports the following operations. (Handle edge cases)

Insertion Operations:
1. Insert a student record at the beginning of the list.
2. Insert a student record at the end of the list.
3. Insert a student record at a specific position (index) in the list.

Deletion Operations:
4. Delete a student record from the beginning of the list.
5. Delete a student record from the end of the list.
6. Delete a student record from a specific position.

Display Operations:
7. Display all student records from start to end.
8. Display all student records from end to start.
```

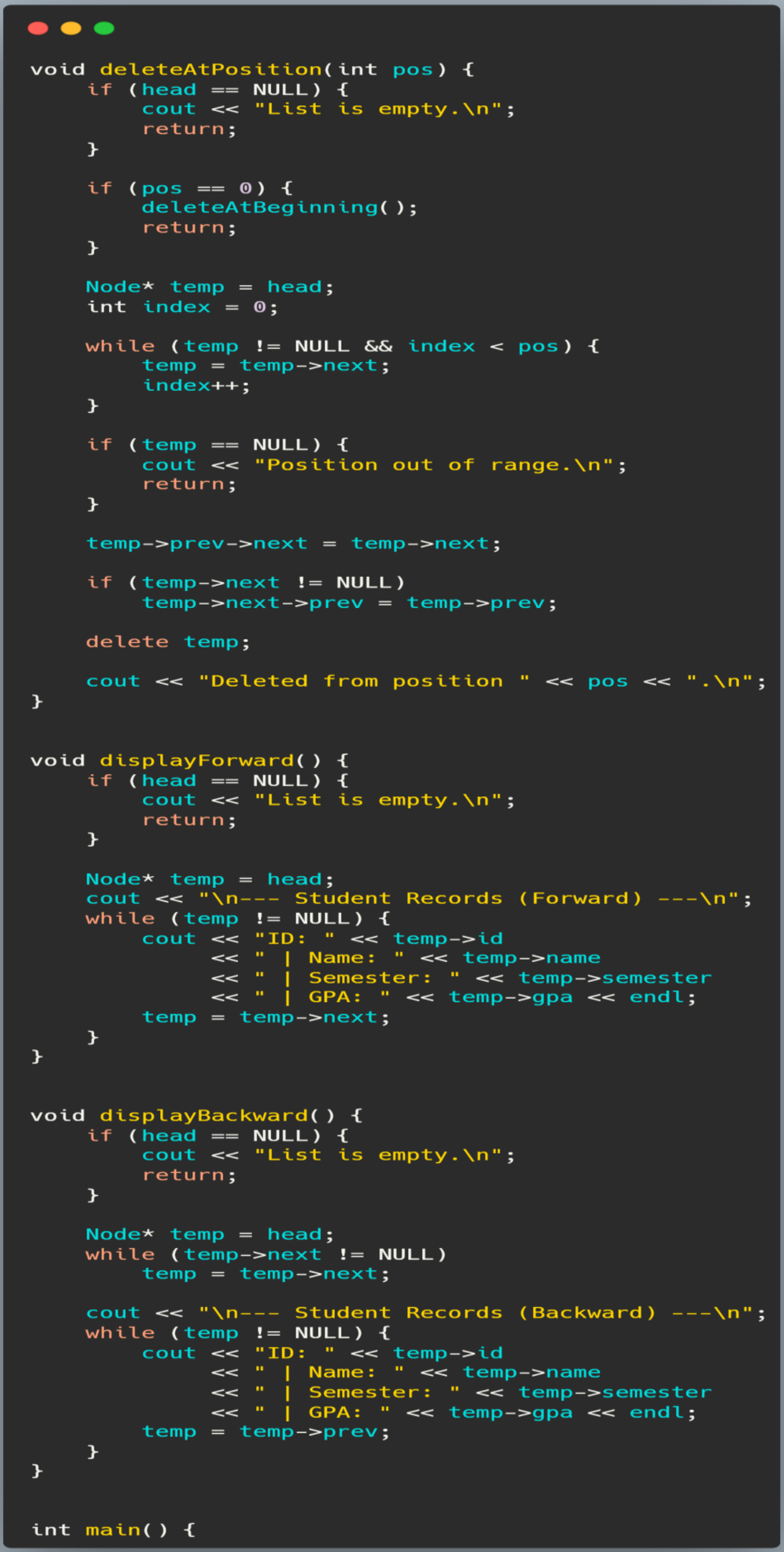
Code:



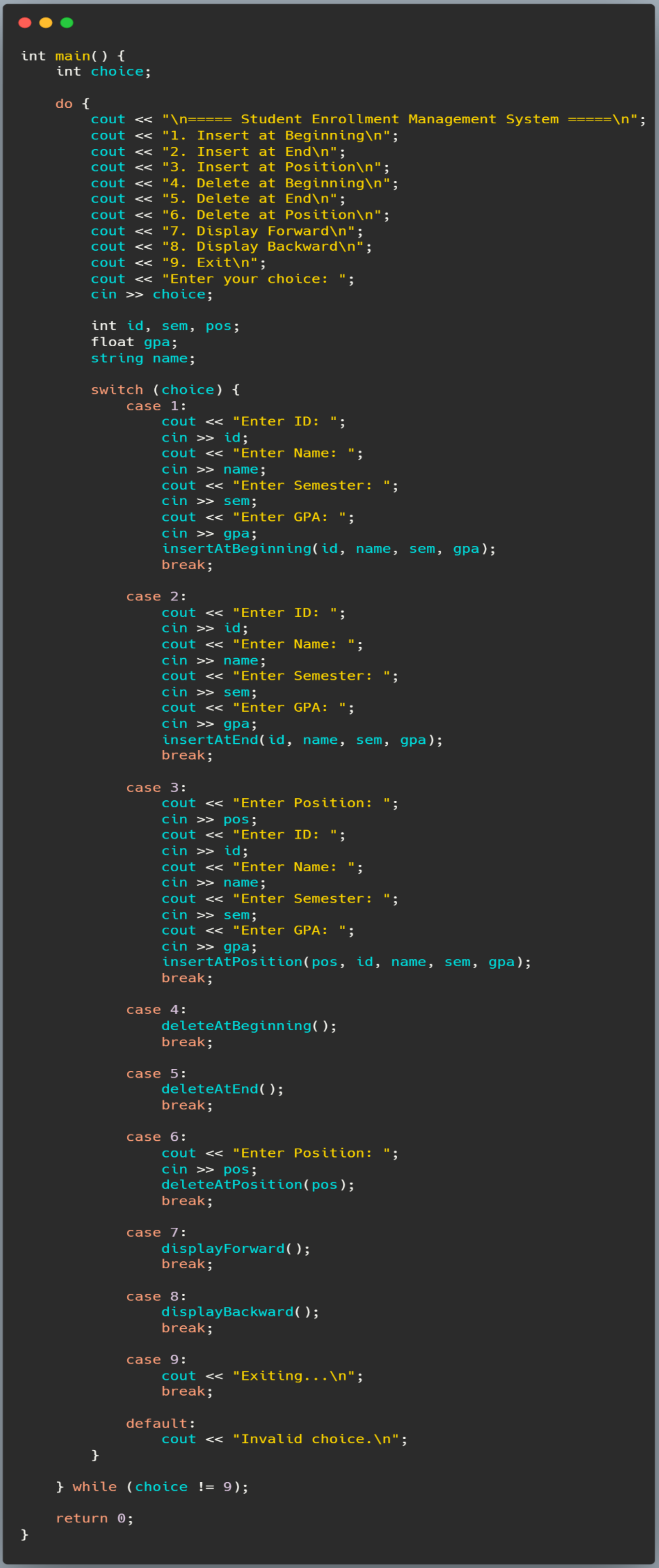
```
#include <iostream>
#include <string>
using namespace std;

struct Node {
    int id;
    string name;
    int semester;
    float gpa;

    Node* next;
    Node* prev;
};

Node* head = NULL;


Node* createNode(int id, string name, int sem, float gpa) {
    Node* n = new Node();
    n->id = id;
    n->name = name;
    n->semester = sem;
    n->gpa = gpa;
    n->next = NULL;
    n->prev = NULL;
    return n;
}


void insertAtBeginning(int id, string name, int sem, float gpa) {
    Node* n = createNode(id, name, sem, gpa);

    if (head == NULL) {
        head = n;
    } else {
        n->next = head;
        head->prev = n;
        head = n;
    }
    cout << "Inserted at beginning.\n";
}


void insertAtEnd(int id, string name, int sem, float gpa) {
    Node* n = createNode(id, name, sem, gpa);

    if (head == NULL) {
        head = n;
        cout << "Inserted at end.\n";
        return;
    }

    Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = n;
    n->prev = temp;

    cout << "Inserted at end.\n";
}


void insertAtPosition(int pos, int id, string name, int sem, float gpa) {
```



```cpp
void insertAtPosition(int pos, int id, string name, int sem, float gpa) {
    if (pos < 0) {
        cout << "Invalid position.\n";
        return;
    }

    if (pos == 0) {
        insertAtBeginning(id, name, sem, gpa);
        return;
    }

    Node* temp = head;
    int index = 0;

    while (temp != NULL && index < pos - 1) {
        temp = temp->next;
        index++;
    }

    if (temp == NULL) {
        cout << "Position out of range.\n";
        return;
    }

    Node* n = createNode(id, name, sem, gpa);
    n->next = temp->next;
    n->prev = temp;

    if (temp->next != NULL)
        temp->next->prev = n;

    temp->next = n;

    cout << "Inserted at position " << pos << ".\n";
}


void deleteAtBeginning() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }

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

    if (head != NULL)
        head->prev = NULL;

    delete temp;
    cout << "Deleted from beginning.\n";
}


void deleteAtEnd() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }

    if (head->next == NULL) {
        delete head;
        head = NULL;
        cout << "Deleted last node.\n";
        return;
    }

    Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    temp->prev->next = NULL;
    delete temp;

    cout << "Deleted from end.\n";
}


void deleteAtPosition(int pos) {
```



```cpp
void deleteAtPosition(int pos) {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }

    if (pos == 0) {
        deleteAtBeginning();
        return;
    }

    Node* temp = head;
    int index = 0;

    while (temp != NULL && index < pos) {
        temp = temp->next;
        index++;
    }

    if (temp == NULL) {
        cout << "Position out of range.\n";
        return;
    }

    temp->prev->next = temp->next;

    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    delete temp;

    cout << "Deleted from position " << pos << ".\n";
}


void displayForward() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }

    Node* temp = head;
    cout << "\n--- Student Records (Forward) ---\n";
    while (temp != NULL) {
        cout << "ID: " << temp->id
             << " | Name: " << temp->name
             << " | Semester: " << temp->semester
             << " | GPA: " << temp->gpa << endl;
        temp = temp->next;
    }
}


void displayBackward() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }

    Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;

    cout << "\n--- Student Records (Backward) ---\n";
    while (temp != NULL) {
        cout << "ID: " << temp->id
             << " | Name: " << temp->name
             << " | Semester: " << temp->semester
             << " | GPA: " << temp->gpa << endl;
        temp = temp->prev;
    }
}

int main() {
```

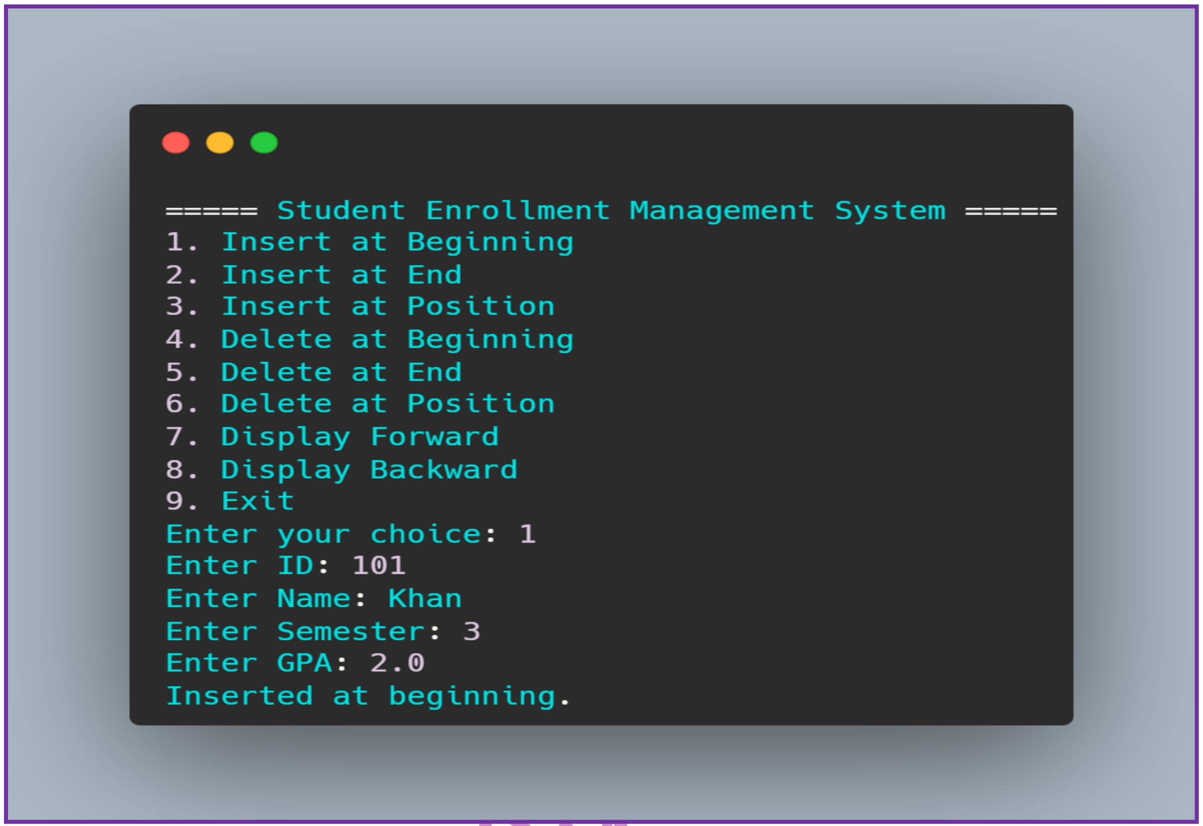


```cpp
int main() {
    int choice;

    do {
        cout << "\n===== Student Enrollment Management System =====\n";
        cout << "1. Insert at Beginning\n";
        cout << "2. Insert at End\n";
        cout << "3. Insert at Position\n";
        cout << "4. Delete at Beginning\n";
        cout << "5. Delete at End\n";
        cout << "6. Delete at Position\n";
        cout << "7. Display Forward\n";
        cout << "8. Display Backward\n";
        cout << "9. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        int id, sem, pos;
        float gpa;
        string name;

        switch (choice) {
            case 1:
                cout << "Enter ID: ";
                cin >> id;
                cout << "Enter Name: ";
                cin >> name;
                cout << "Enter Semester: ";
                cin >> sem;
                cout << "Enter GPA: ";
                cin >> gpa;
                insertAtBeginning(id, name, sem, gpa);
                break;

            case 2:
                cout << "Enter ID: ";
                cin >> id;
                cout << "Enter Name: ";
                cin >> name;
                cout << "Enter Semester: ";
                cin >> sem;
                cout << "Enter GPA: ";
                cin >> gpa;
                insertAtEnd(id, name, sem, gpa);
                break;

            case 3:
                cout << "Enter Position: ";
                cin >> pos;
                cout << "Enter ID: ";
                cin >> id;
                cout << "Enter Name: ";
                cin >> name;
                cout << "Enter Semester: ";
                cin >> sem;
                cout << "Enter GPA: ";
                cin >> gpa;
                insertAtPosition(pos, id, name, sem, gpa);
                break;

            case 4:
                deleteAtBeginning();
                break;

            case 5:
                deleteAtEnd();
                break;

            case 6:
                cout << "Enter Position: ";
                cin >> pos;
                deleteAtPosition(pos);
                break;

            case 7:
                displayForward();
                break;

            case 8:
                displayBackward();
                break;

            case 9:
                cout << "Exiting...\n";
                break;

            default:
                cout << "Invalid choice.\n";
        }

    } while (choice != 9);

    return 0;
}
```

Table Of Output:

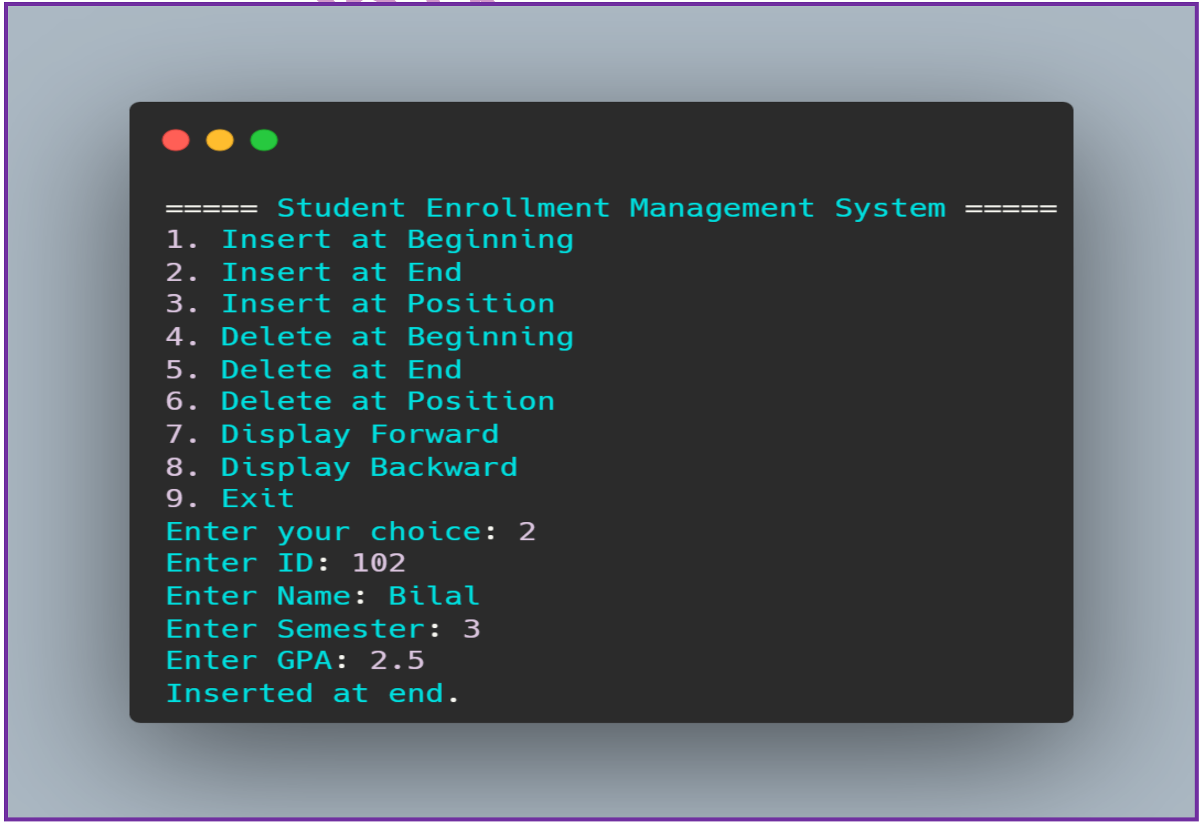
	Condition	Output
1	Insert at beginning	Inserted at beginning.
2	Insert at end (empty or non-empty list)	Inserted at end.
3	Insert at position (valid)	Inserted at position X.
4	Insert at position (pos < 0)	Invalid position.
5	Insert at position (out of range)	Position out of range.
6	Delete at beginning (list empty)	List is empty.
7	Delete at beginning (list not empty)	Deleted from beginning.
8	Delete at end (list empty)	List is empty.
9	Delete at end (single node)	Deleted last node.
10	Delete at end (multiple nodes)	Deleted from end.
11	Delete at position (list empty)	List is empty.
12	Delete at position (out of range)	Position out of range.
13	Delete at position (valid)	Deleted from position X.
14	Display forward (list empty)	List is empty.
15	Display forward (list not empty)	Student records printed (forward)
16	Display backward (list empty)	List is empty.
17	Display backward (list not empty)	Student records printed (backward)
18	Exit selected	Exiting...
19	Invalid menu choice	Invalid choice.
	Total 19 Possible Outputs	

When Insert at beginning:



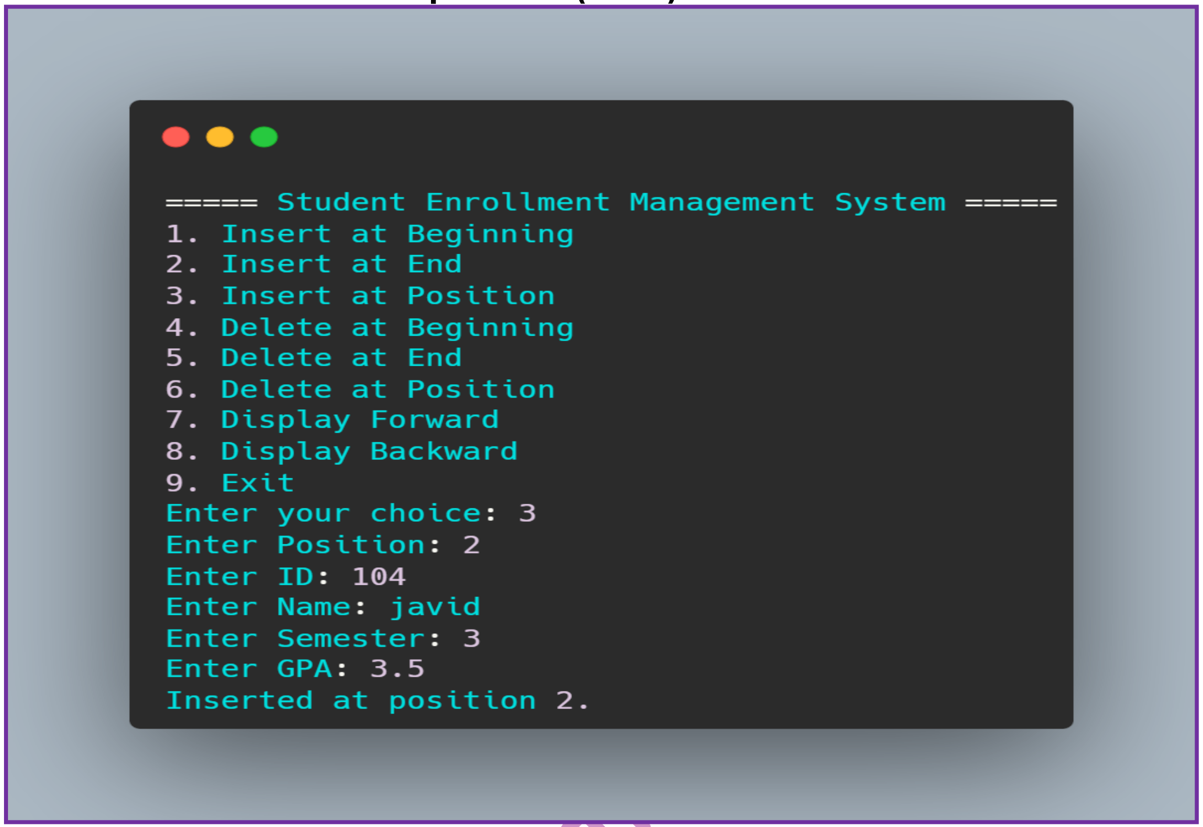
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 1
Enter ID: 101
Enter Name: Khan
Enter Semester: 3
Enter GPA: 2.0
Inserted at beginning.
```

When Insert at end (empty or non-empty list):



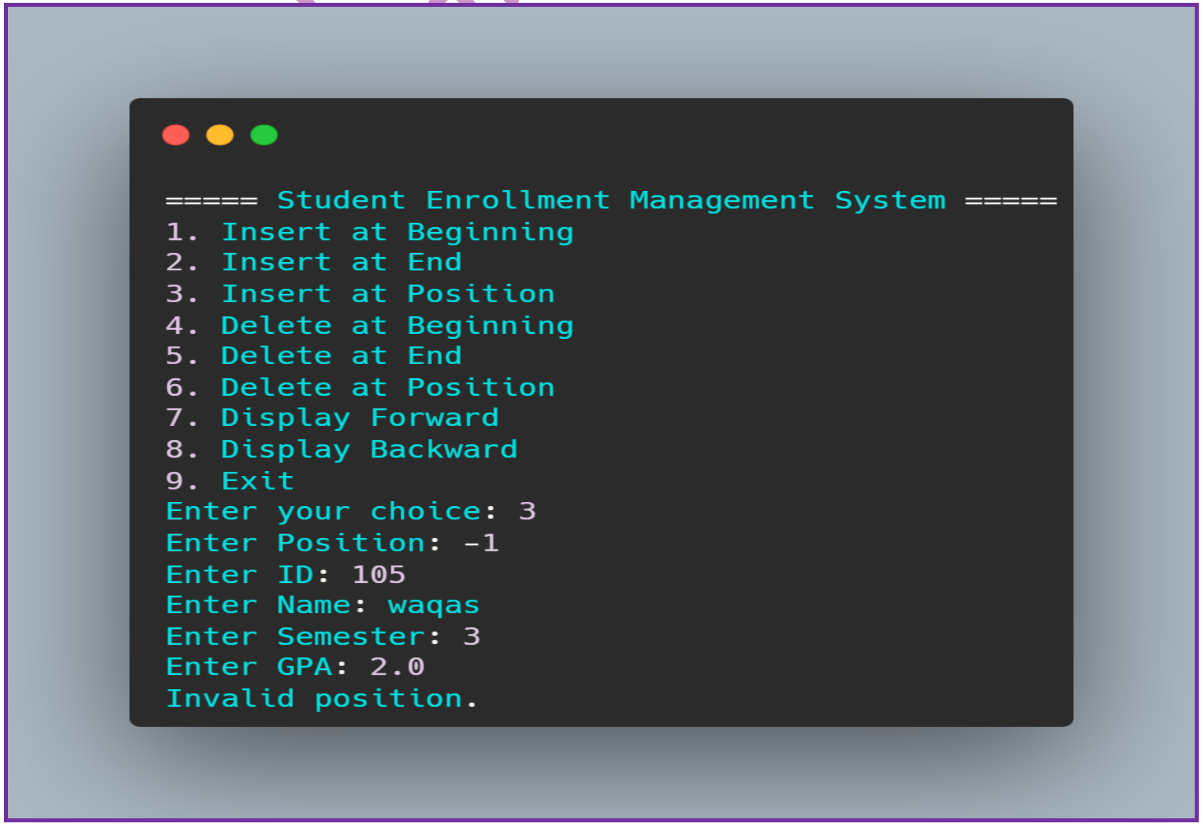
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 2
Enter ID: 102
Enter Name: Bilal
Enter Semester: 3
Enter GPA: 2.5
Inserted at end.
```

When Insert at position (valid):



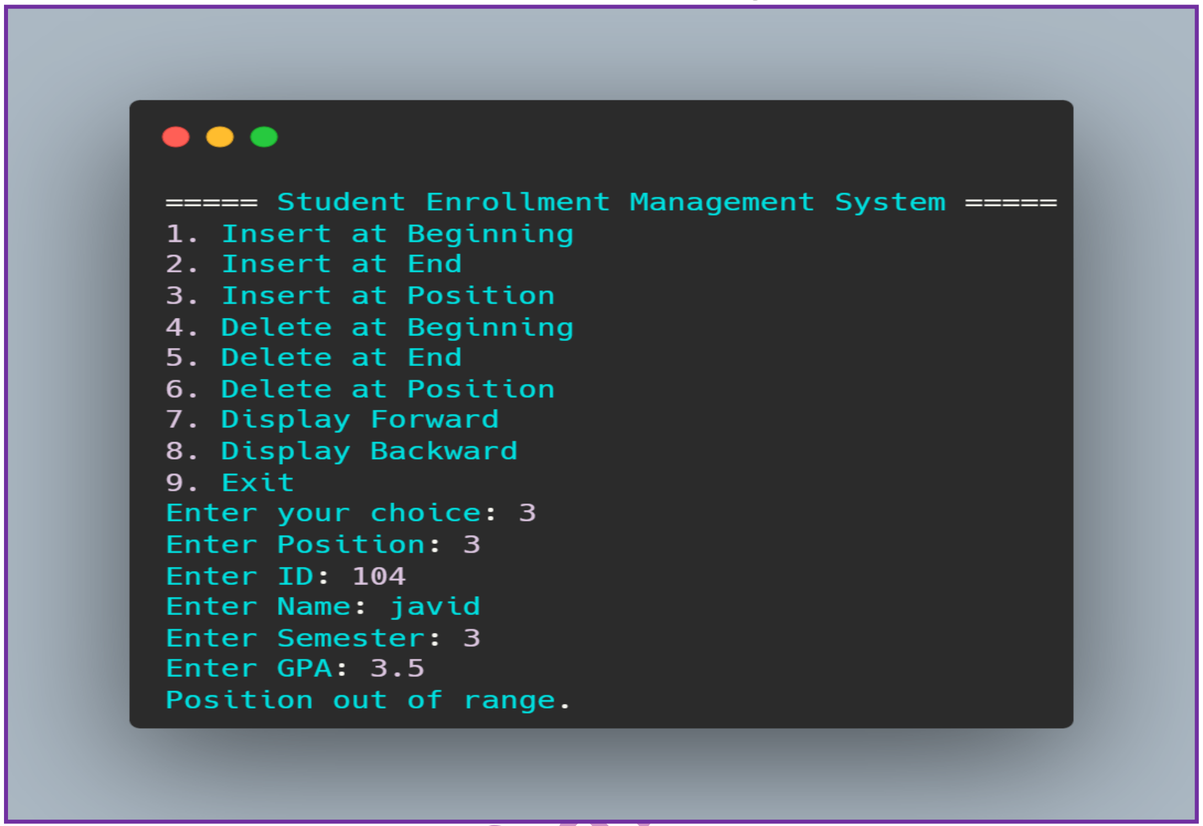
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 3
Enter Position: 2
Enter ID: 104
Enter Name: javid
Enter Semester: 3
Enter GPA: 3.5
Inserted at position 2.
```

When Insert at position (pos < 0):



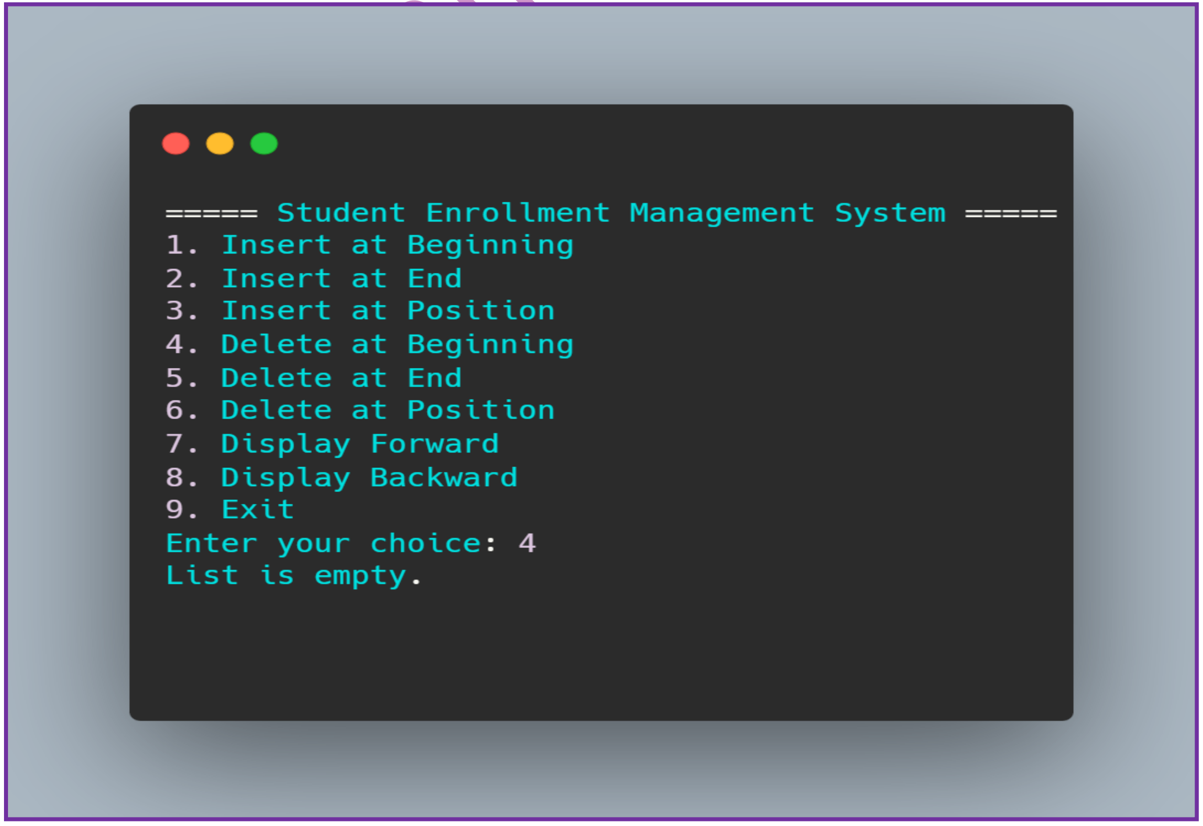
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 3
Enter Position: -1
Enter ID: 105
Enter Name: waqas
Enter Semester: 3
Enter GPA: 2.0
Invalid position.
```

When Insert at position (out of range):



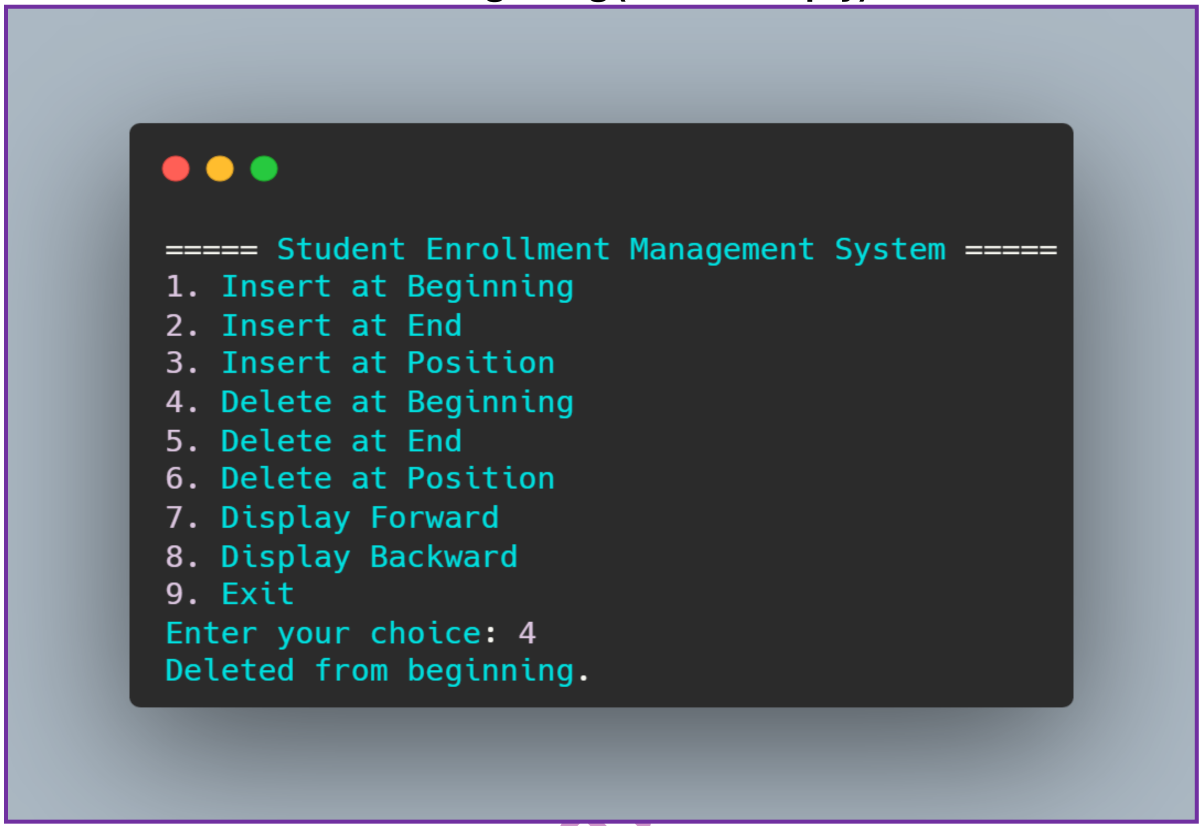
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 3
Enter Position: 3
Enter ID: 104
Enter Name: javid
Enter Semester: 3
Enter GPA: 3.5
Position out of range.
```

When Delete at beginning (list empty):



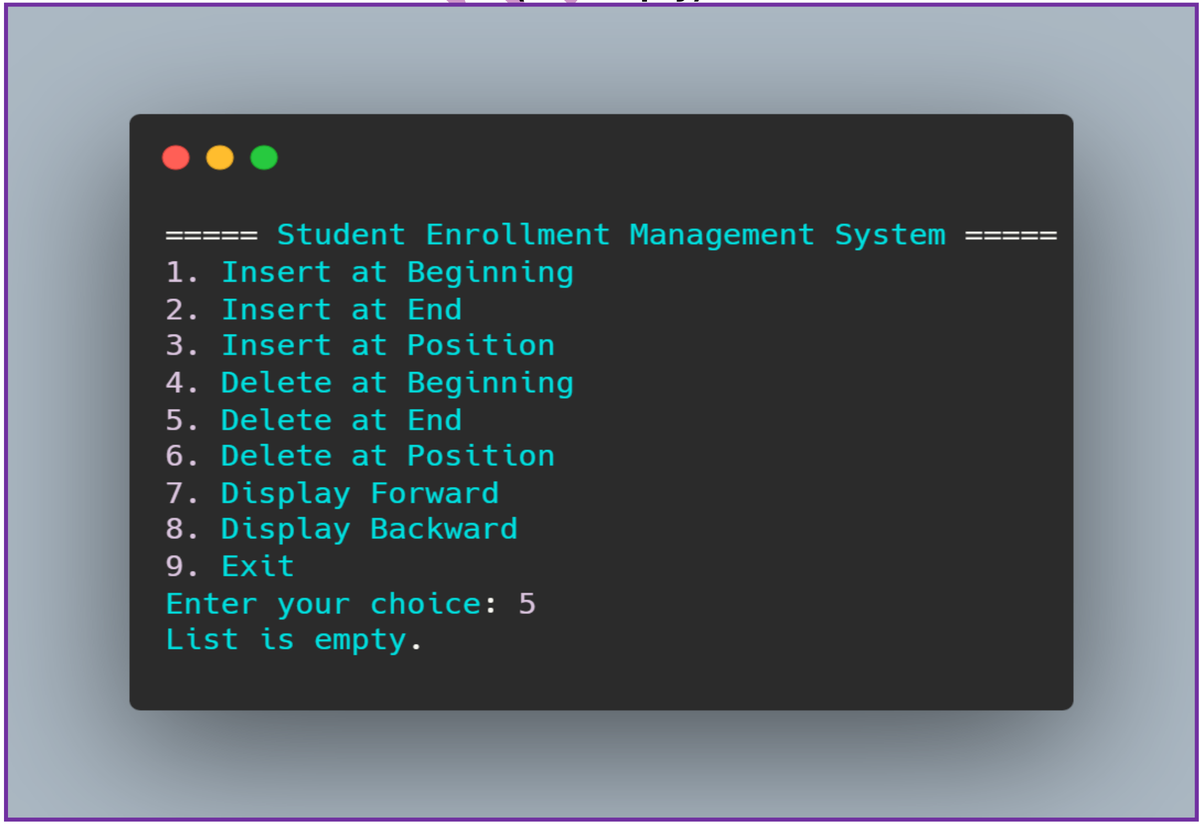
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 4
List is empty.
```

When **Delete at beginning (list not empty):**



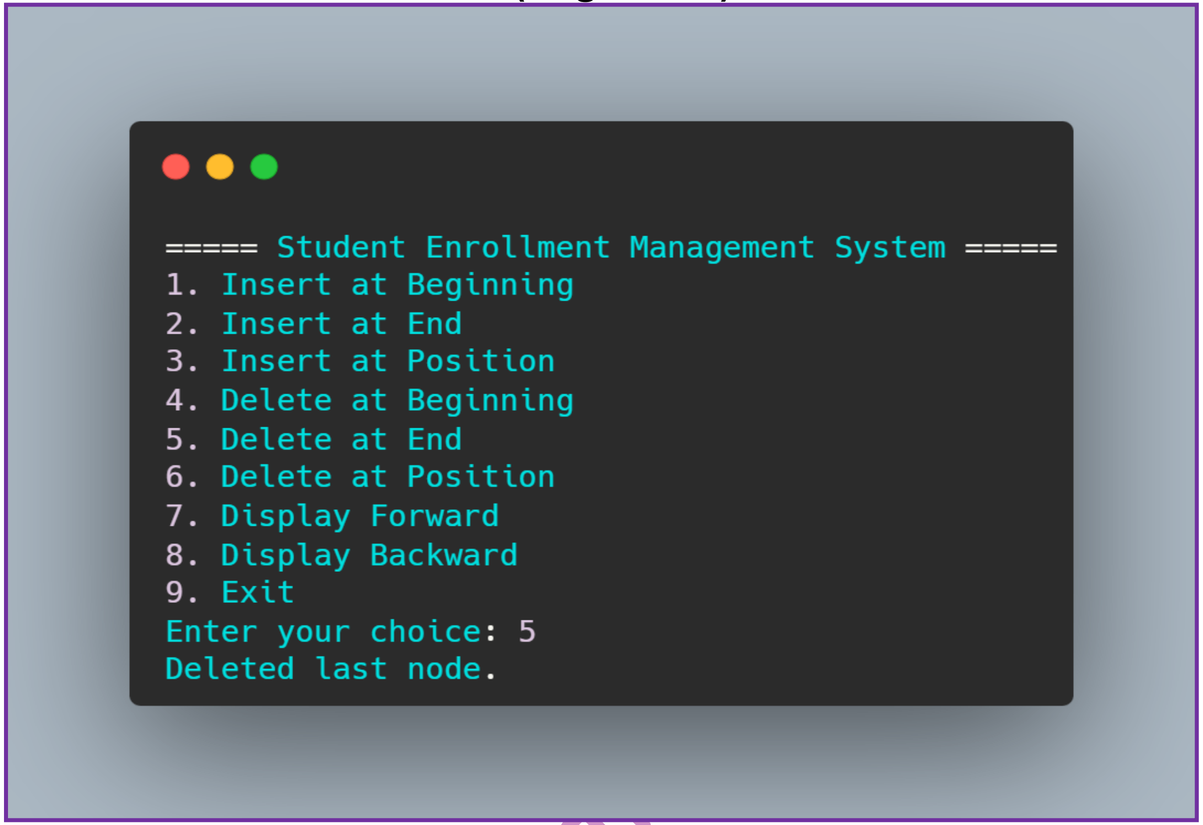
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 4
Deleted from beginning.
```

When **Delete at end (list empty):**



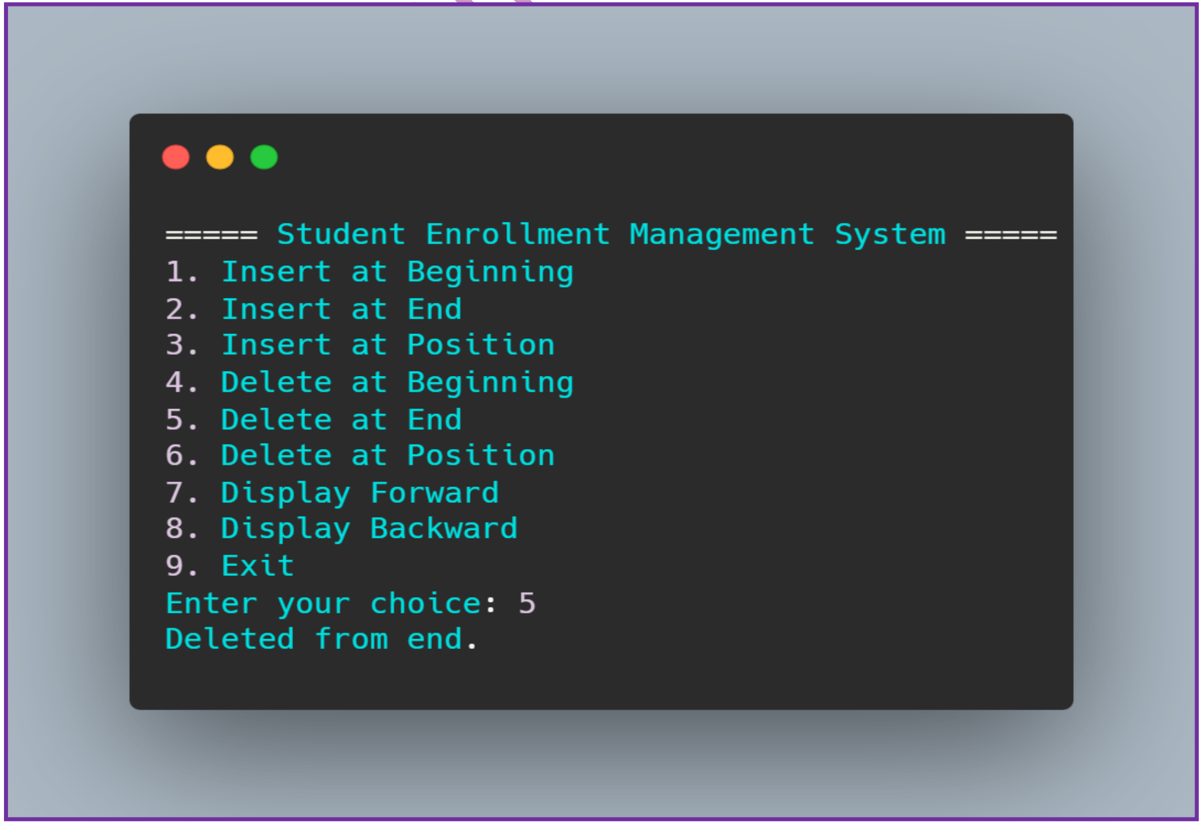
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 5
List is empty.
```

When **Delete at end (single node):**



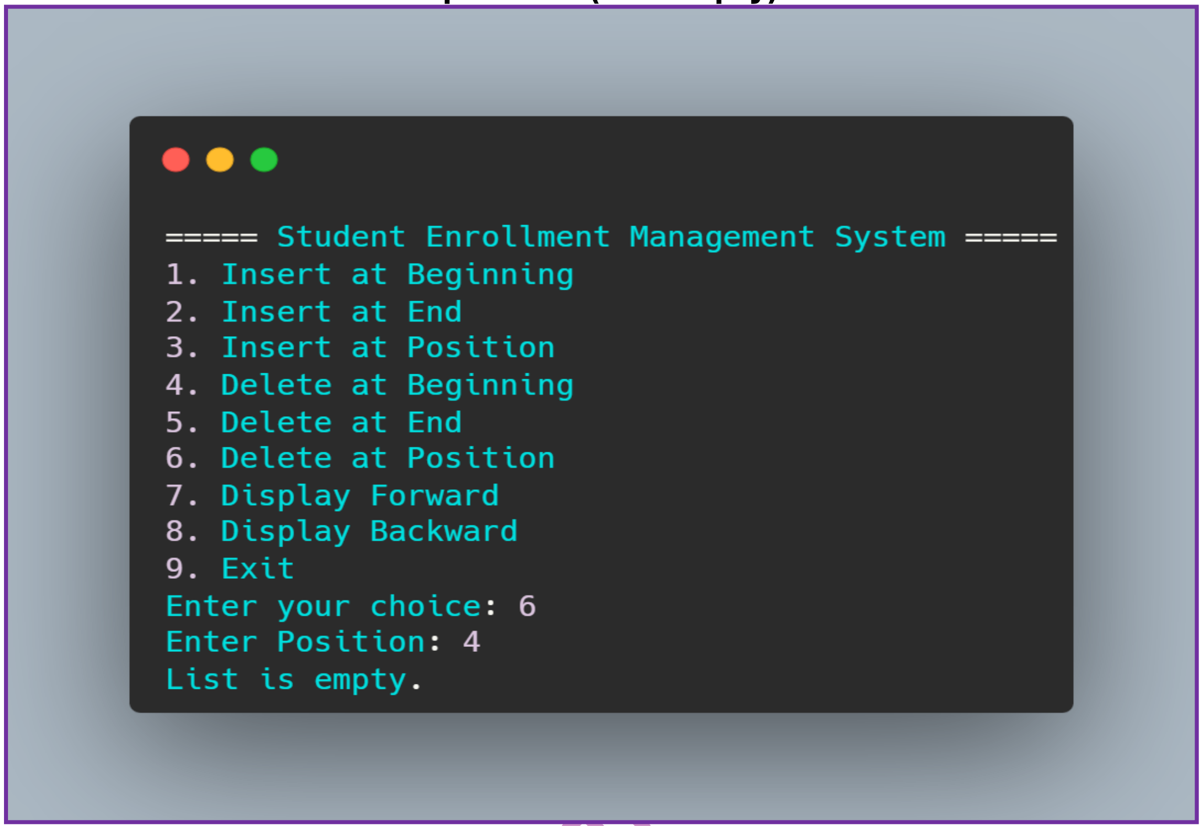
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 5
Deleted last node.
```

When **Delete at end (multiple nodes):**



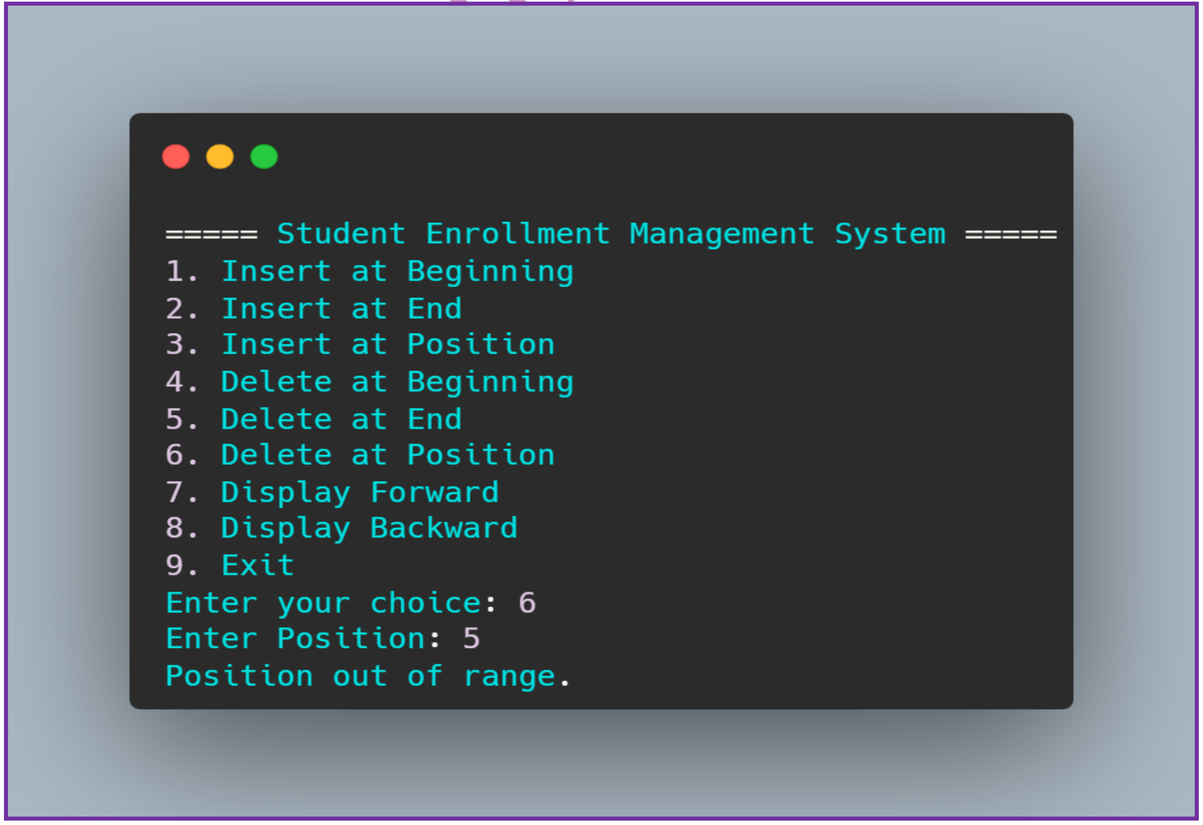
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 5
Deleted from end.
```

When **Delete at position (list empty):**



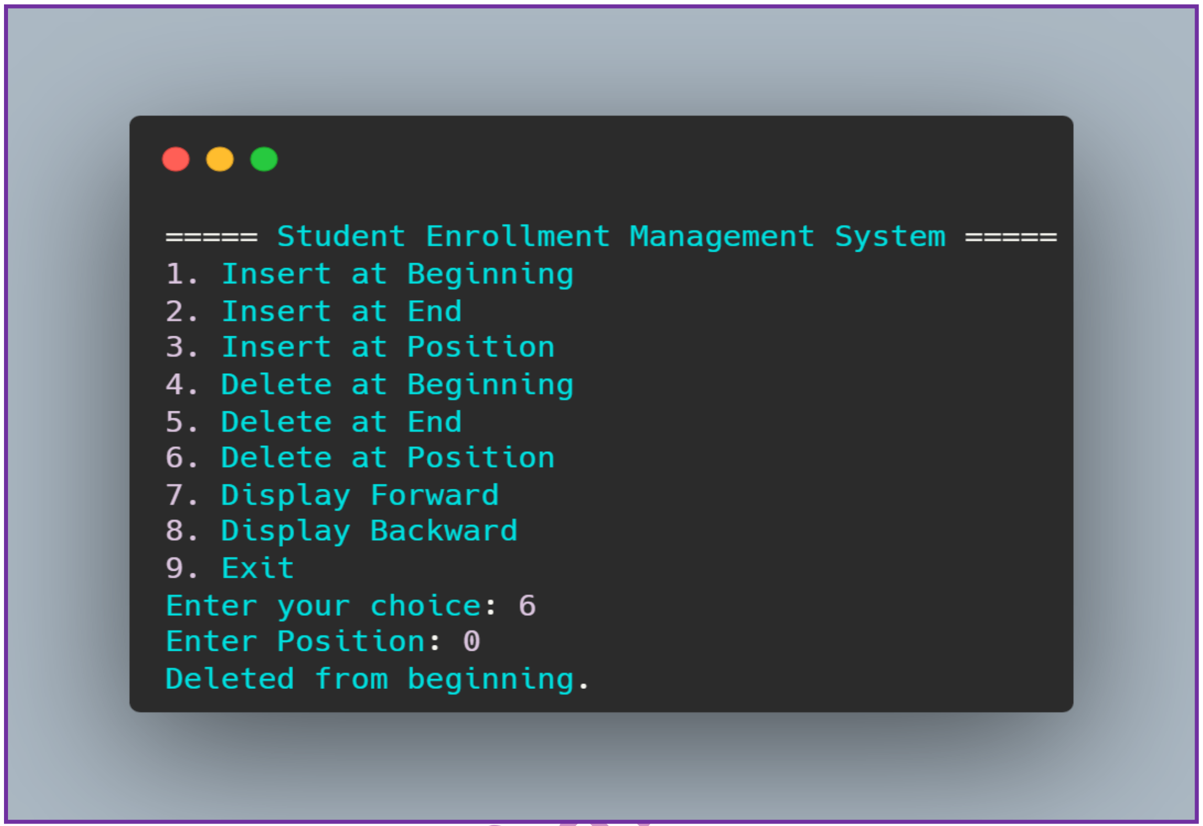
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 6
Enter Position: 4
List is empty.
```

When **Delete at position (out of range):**



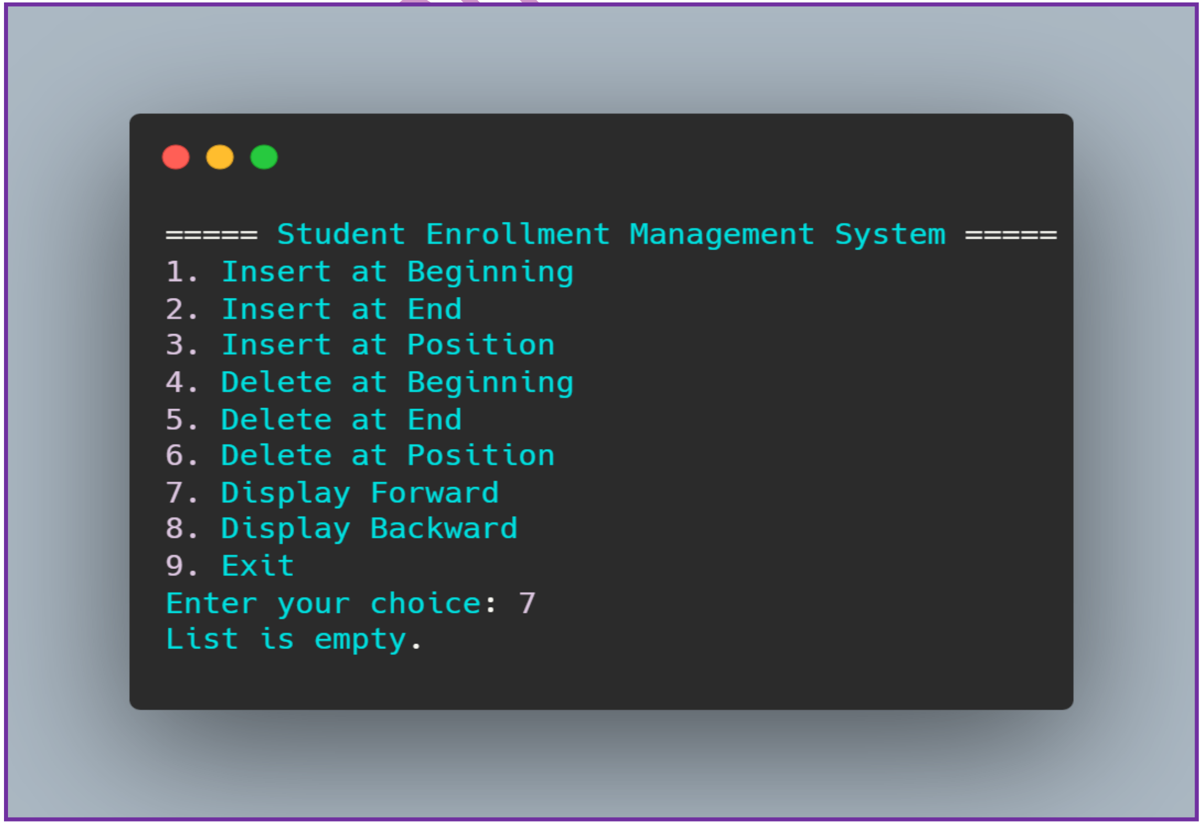
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 6
Enter Position: 5
Position out of range.
```

When Delete at position (valid):



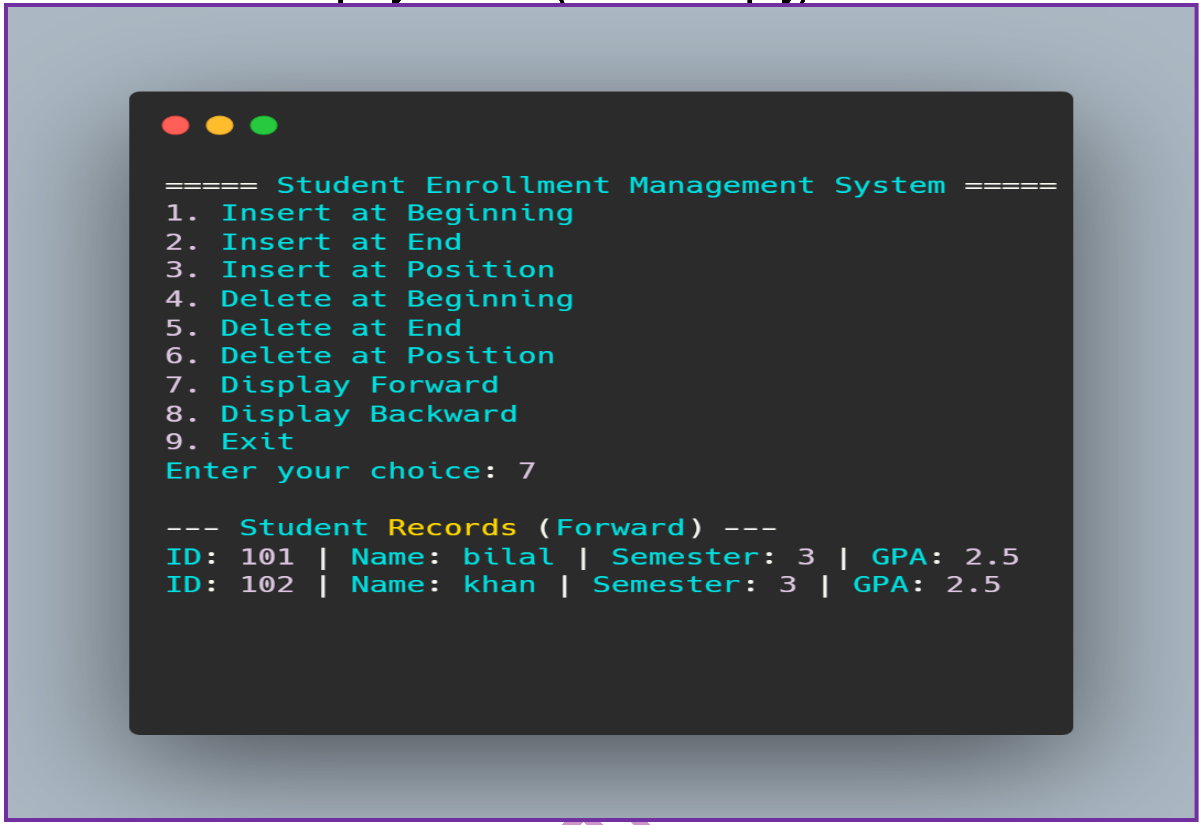
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 6
Enter Position: 0
Deleted from beginning.
```

When Display forward (list empty):



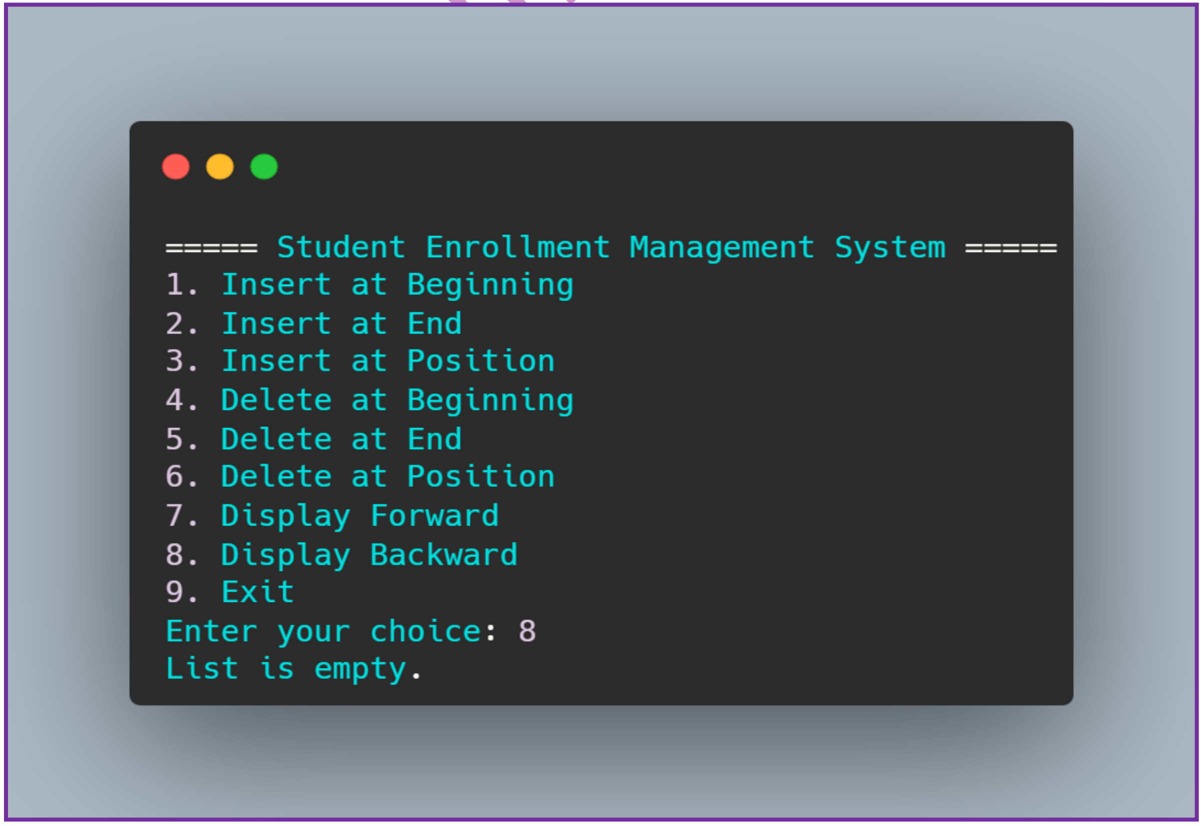
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 7
List is empty.
```

When Display forward (list not empty):



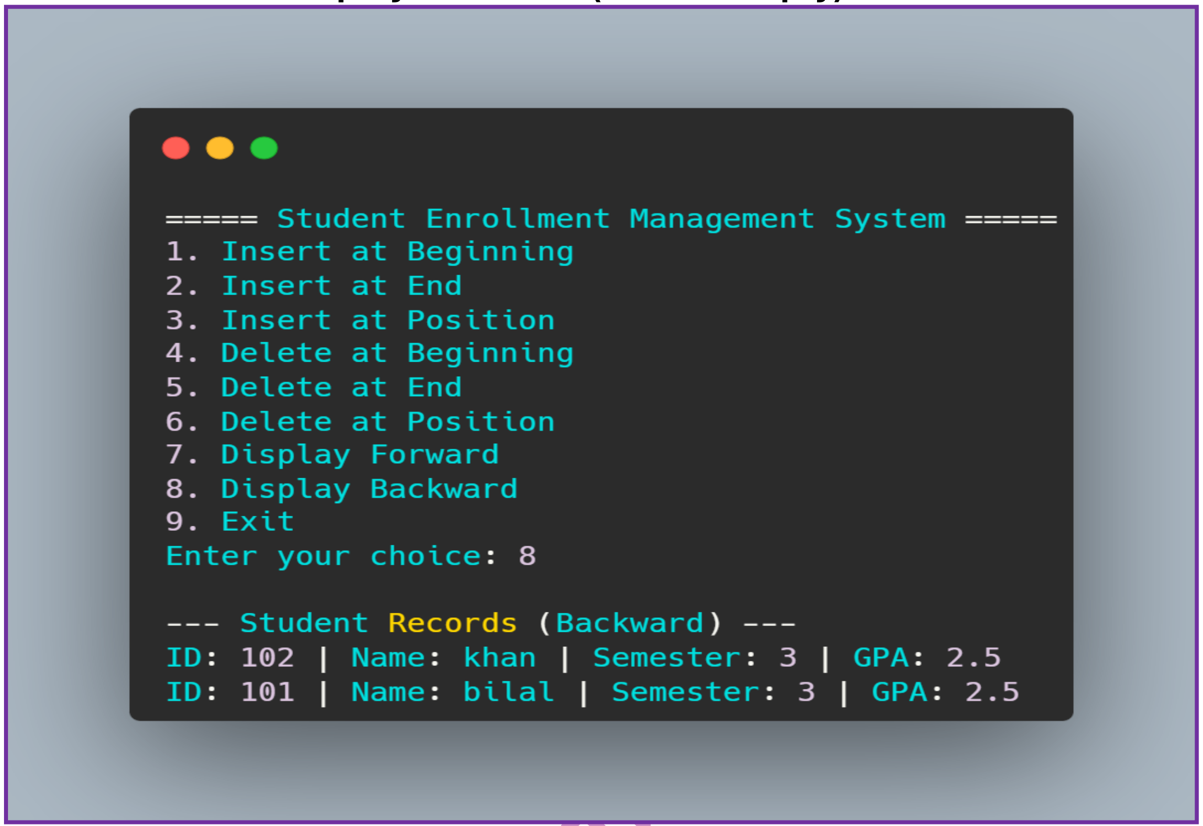
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 7

--- Student Records (Forward) ---
ID: 101 | Name: bilal | Semester: 3 | GPA: 2.5
ID: 102 | Name: khan | Semester: 3 | GPA: 2.5
```

When Display backward (list empty):



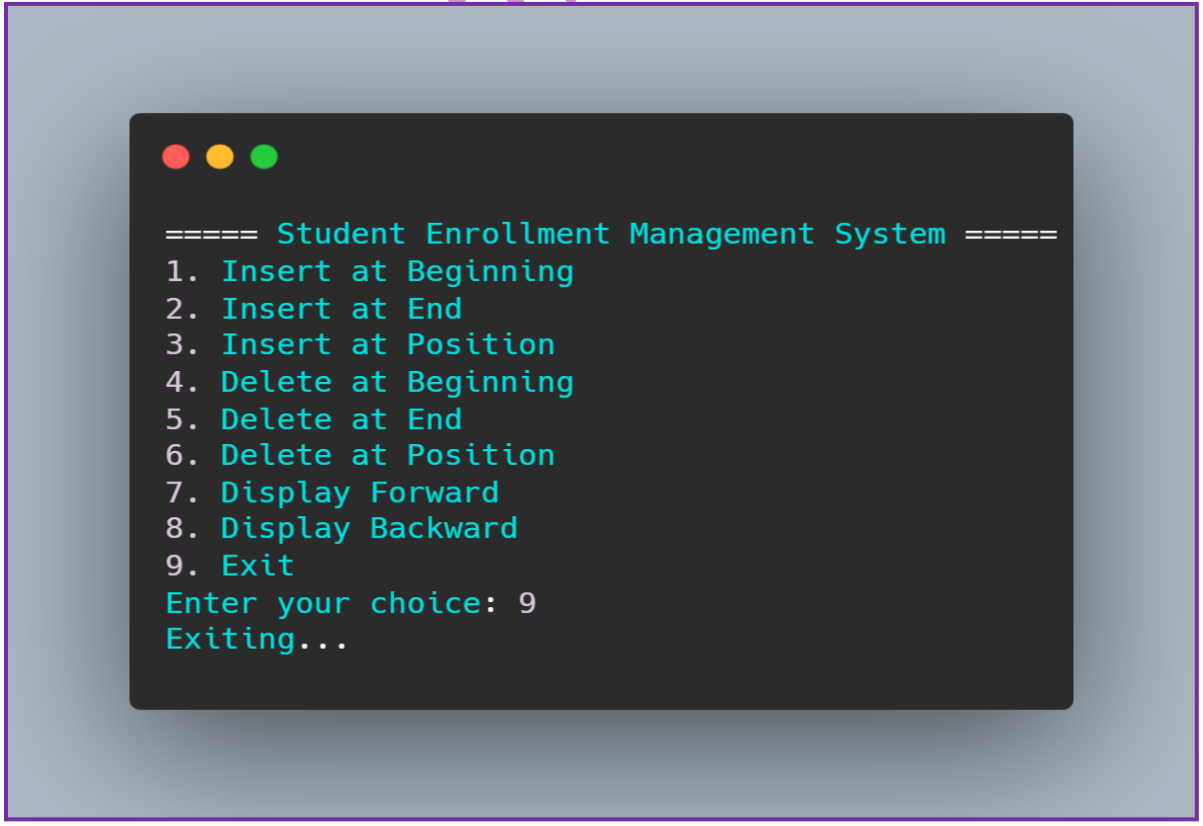
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 8
List is empty.
```

When Display backward (list not empty):



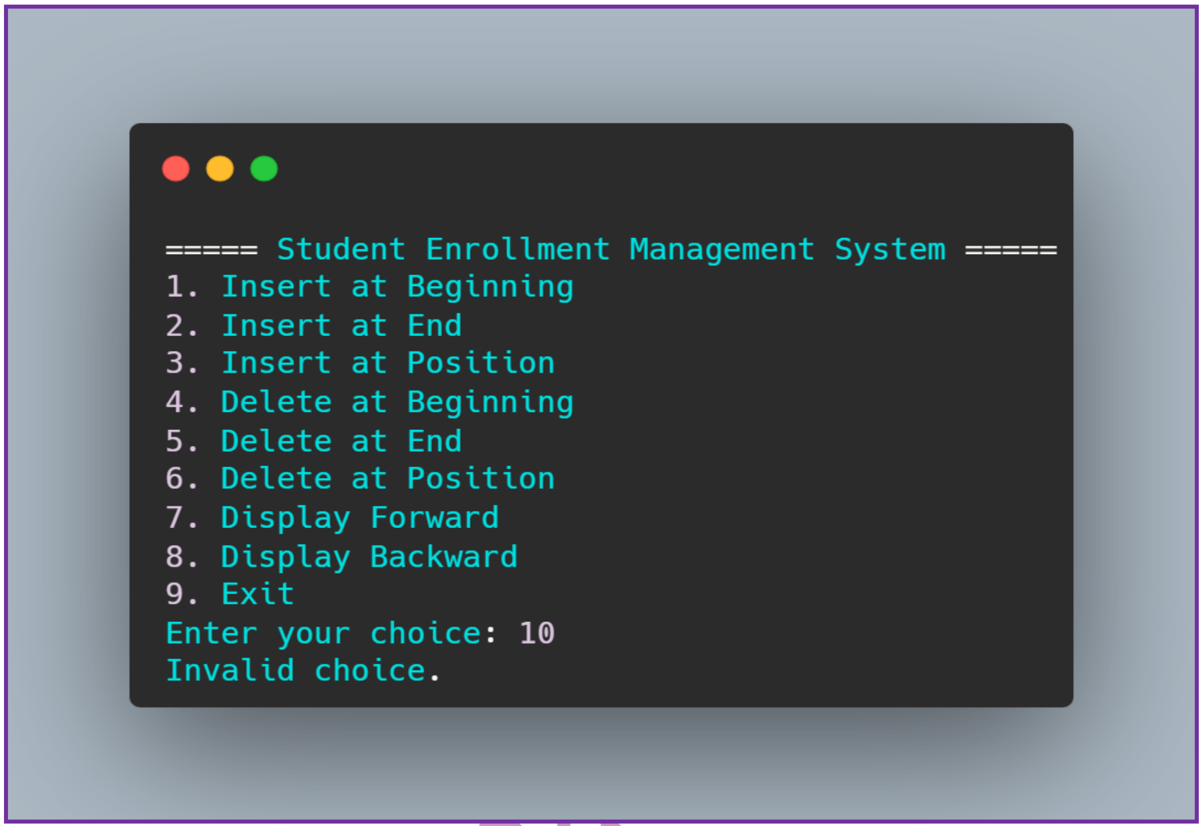
```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 8

--- Student Records (Backward) ---
ID: 102 | Name: khan | Semester: 3 | GPA: 2.5
ID: 101 | Name: bilal | Semester: 3 | GPA: 2.5
```

When Exit selected:



```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 9
Exiting...
```

When Invalid menu choice:



```
===== Student Enrollment Management System =====
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Delete at Beginning
5. Delete at End
6. Delete at Position
7. Display Forward
8. Display Backward
9. Exit
Enter your choice: 10
Invalid choice.
```

Click here to Get this code on GitHub

Click here to Test this Code by Yourself.