LINKED LIST IMPLEMENTATION

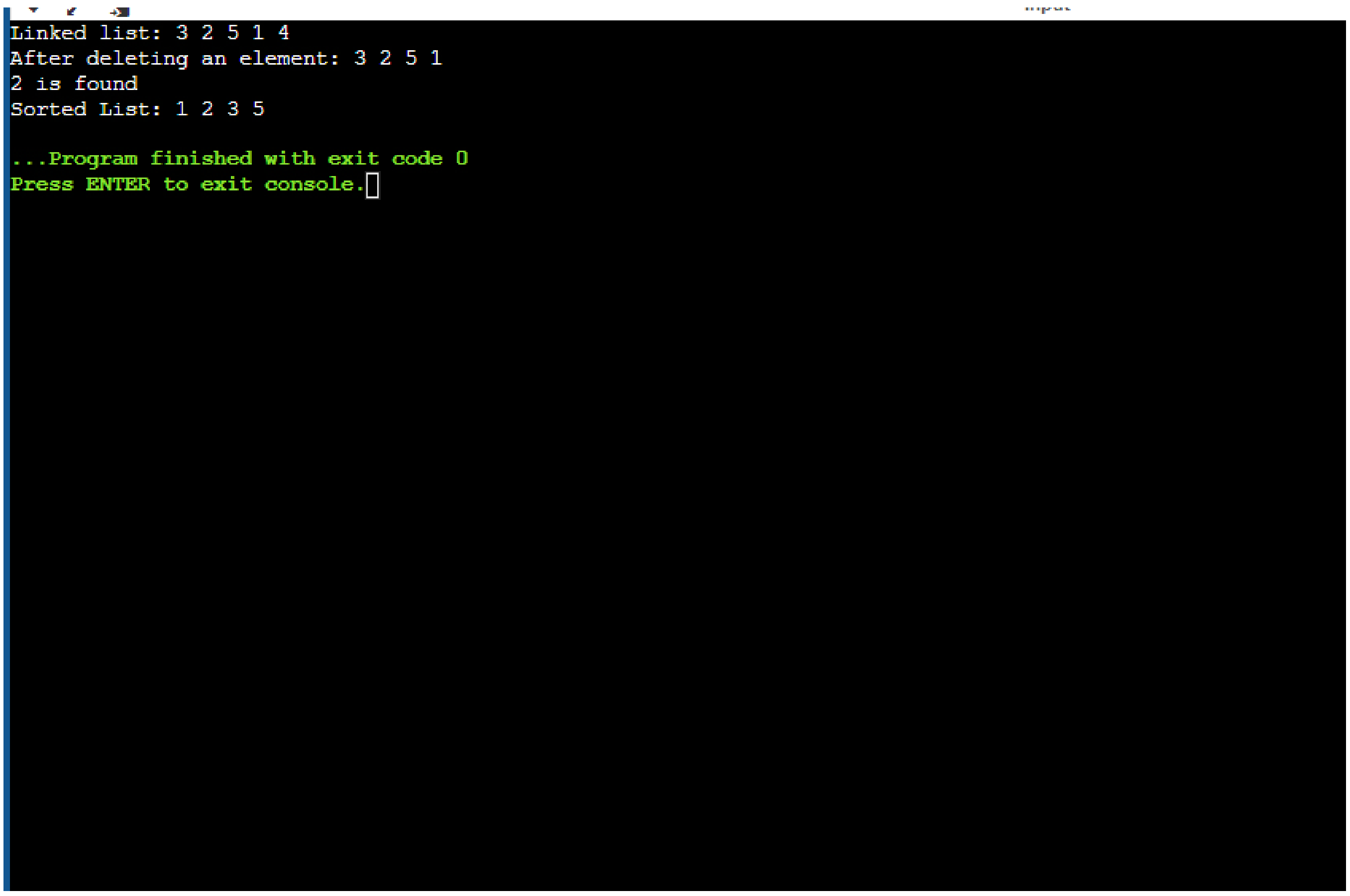
CODE-

```
#include <bits/stdc++.h>
using namespace std;
struct Node {
  int data;
  struct Node* next;
};

void insertAtBeginning(struct Node** head_ref, int new_data) {
  // Allocate memory to a node
  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

  // insert the data
  new_node->data = new_data;
  new_node->next = (*head_ref);

  // Move head to new node
  (*head_ref) = new_node;
}

// Insert a node after a node
void insertAfter(struct Node* prev_node, int new_data) {
  if (prev_node == NULL) {
  cout << "the given previous node cannot be NULL";
  return;
  }

  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
  new_node->data = new_data;
  new_node->next = prev_node->next;
  prev_node->next = new_node;
}

// Insert at the end
void insertAtEnd(struct Node** head_ref, int new_data) {
  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
  struct Node* last = *head_ref; /* used in step 5*/

  new_node->data = new_data;
  new_node->next = NULL;

  if (*head_ref == NULL) {
  *head_ref = new_node;
  return;
  }

  while (last->next != NULL) last = last->next;

  last->next = new_node;
  return;
}

// Delete a node
void deleteNode(struct Node** head_ref, int key) {
  struct Node *temp = *head_ref, *prev;

  if (temp != NULL && temp->data == key) {
  *head_ref = temp->next;
```

```
 }

  free(temp);
  return;
  }
  // Find the key to be deleted
  while (temp != NULL && temp->data != key) {
  prev = temp;
  temp = temp->next;
  }

  // If the key is not present
  if (temp == NULL) return;

  // Remove the node
  prev->next = temp->next;

  free(temp);
}

// Search a node
bool searchNode(struct Node** head_ref, int key) {
  struct Node* current = *head_ref;

  while (current != NULL) {
  if (current->data == key) return true;
  current = current->next;
  }
  return false;
}

// Sort the linked list
```

```
void sortLinkedList(struct Node** head_ref) {
  struct Node *current = *head_ref, *index = NULL;
  int temp;

  if (head_ref == NULL) {
  return;
  } else {
  while (current != NULL) {
    // index points to the node next to current
    index = current->next;

    while (index != NULL) {
    if (current->data > index->data) {
      temp = current->data;
      current->data = index->data;
      index->data = temp;
    }
    index = index->next;
    }
    current = current->next;
  }
  }
}

// Print the linked list
void printList(struct Node* node) {
  while (node != NULL) {
  cout << node->data << " ";
  node = node->next;
  }
}

// Driver program
int main() {
  struct Node* head = NULL;

  insertAtEnd(&head, 1);
  insertAtBeginning(&head, 2);
  insertAtBeginning(&head, 3);
  insertAtEnd(&head, 4);
  insertAfter(head->next, 5);

  cout << "Linked list: ";
  printList(head);

  cout << "\nAfter deleting an element: ";
  deleteNode(&head, 4);
  printList(head);

  int item_to_find = 2;
  if (searchNode(&head, item_to_find)) {
  cout << endl << item_to_find << " is found";
  } else {
  cout << endl << item_to_find << " is not found";
  }

  sortLinkedList(&head);
  cout << "\nSorted List: ";
  printList(head);
}
```

OUTPUT-



```
Linked list: 3 2 5 1 4
After deleting an element: 3 2 5 1
2 is found
Sorted List: 1 2 3 5

...Program finished with exit code 0
Press ENTER to exit console.
```