

The screenshot shows the Code::Blocks IDE interface with the following details:

- Title Bar:** AndyLinkedit.C - Code::Blocks 20.03
- Menu Bar:** File Edit View Search Project Build Debug Fortran wsSmith Tools Plugins DocBlocks Settings Help
- Code Editor:** The main window displays C code for creating a linked list. The code includes header file includes, a struct definition for a node, and functions for creating a list and inserting data at the beginning.
- Toolbars:** Standard toolbar icons for file operations like Open, Save, Print, and Run are visible along the top.
- Status Bar:** Shows the current file path as AndyLinkedit.C and the line number 1.

```
1 //include <csio.h>
2 //include <stdlib.h>
3
4 struct node {
5     int data;
6     struct node *next;
7 }
8
9 struct node *head = NULL;
10
11
12
13 void createList(int n) {
14     struct node *newNode, *temp = NULL;
15     int data, i;
16
17     if (n <= 0)
18         printf("Number of nodes should be greater than 0\n");
19     return;
20 }
21
22
23 if (head == NULL) {
24     printf("Warning: Overwriting existing list.\n");
25     head = NULL;
26 }
27
28
29 for (i = 1; i <= n; i++) {
30     newNode = (struct node*)malloc(sizeof(struct node));
31     if (newNode == NULL) {
32         printf("Memory allocation failed\n");
33         return;
34     }
35     printf("Enter data for node %d: ", i);
36     scanf("%d", &data);
37
38     newNode->data = data;
39     newNode->next = NULL;
40
41     if (head == NULL)
42         head = newNode;
43     else
44         temp->next = newNode;
45
46     temp = newNode;
47 }
48
49 printf("\nLinked list created successfully\n");
50
51
52 void insertAtBeginning(int data) {
53     struct node *newNode = (struct node*)malloc(sizeof(struct node));
54     if (newNode == NULL) {
```

The screenshot shows the Code::Blocks IDE interface with a C project named "singlylinkedlist.c". The code implements a singly linked list with functions for inserting nodes at the beginning, end, and a specified position.

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure of a node
struct node {
    int data;
    struct node *next;
};

// Function to insert a new node at the beginning of the list
void insertAtBeginning(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("Node inserted at the beginning\n");
}

// Function to insert a new node at the end of the list
void insertAtEnd(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
    } else {
        struct node *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
    printf("Node inserted at the end\n");
}

// Function to insert a new node at a specific position
void insertAtPosition(int data, int pos) {
    int i;
    struct node *newNode, *temp = head;
    if (pos < 1) {
        printf("Invalid position. Position must be 1 or greater.\n");
        return;
    }
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    for (i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;
}
```

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};

void insertAtBeginning(int data) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}

void insertAtEnd(int data) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    struct node *temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void insertAtPosition(int data, int pos) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    struct node *temp = head;
    int count = 1;
    while (temp != NULL && count < pos) {
        temp = temp->next;
        count++;
    }
    if (temp == NULL) {
        printf("Position out of range: List is not long enough to reach position %d.\n", pos);
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

void displayList() {
    struct node *temp = head;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("\nLinked list: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\nNULL\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at any Position\n");
        printf("4. Insert at End\n");
        printf("5. Display linked list\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if (choice > 6 || choice < 1) {
            while (getchar() != '\n') {
                printf("Invalid input. Please enter a number.\n");
            }
            continue;
        }
        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &pos);
                insertAtPosition(data, pos);
                break;
            case 4:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                displayList();
                break;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};

void insertAtBeginning(int data) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}

void insertAtEnd(int data) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    struct node *temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

void insertAtPosition(int data, int pos) {
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    struct node *temp = head;
    int count = 1;
    while (temp != NULL && count < pos) {
        temp = temp->next;
        count++;
    }
    if (temp == NULL) {
        printf("Position out of range: List is not long enough to reach position %d.\n", pos);
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

void displayList() {
    struct node *temp = head;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("\nLinked list: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\nNULL\n");
}

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at any Position\n");
        printf("4. Insert at End\n");
        printf("5. Display linked list\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        if (choice > 6 || choice < 1) {
            while (getchar() != '\n') {
                printf("Invalid input. Please enter a number.\n");
            }
            continue;
        }
        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &pos);
                insertAtPosition(data, pos);
                break;
            case 4:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                displayList();
                break;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }
    return 0;
}
```

```
D:\Dhruv\PycharmProjects\linkedlist>python singlylinkedlist.py
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 1
Enter number of nodes: 2
Enter data for node 1: 3
Enter data for node 2: 4
linked list created successfully
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 5
linked list: 3->4->NULL
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 2
Enter data to insert: 344
node inserted at the beginning
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 5
linked list: 344->3->4->NULL
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 4
Enter data to insert: 878
node inserted at the end
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
```

- 1) Write a program to implement singly linked list with
 the following
 a) Create a linked list
 b) Insertion at a node at
 first position & any position & end of the list
 c) Display the contents of linked list

pseudocode

structure node

data

next → node

end structure

→ empty linked list = head ← null

→ New node creation

create node (value)

Newnode ← allocate memory for node

Newnode.data ← value

Newnode.next ← null

return node

end function

→ Insert at first position

firstposition (value)

Newnode ← create node (value)

Newnode.next ← head

head ← Newnode

end function

→ Insert at end

insertatend (value)

Newnode → create node (value)

if head = null then

head ← Newnode

return



Shot on OnePlus
chE_Reddy ← head

DATE:

function insert (head, targetvalue, newvalue)
{
 if (temp.next == null)
 temp = temp.next
 else
 temp = temp.next
 temp.next = newnode
 newnode.next = temp.next
 return head
}

⇒ Insert at Any position

InsertAtAnyPosition (targetvalue, newvalue)

temp ← head

while temp ≠ null and temp.data ≠ targetvalue
 temp = temp.next

end while

if temp = null then

print "target not found", newvalue

return null

endif

newnode ← createNode (newvalue)

newnode.next ← temp.next

temp.next ← newnode

end

⇒ Display

Display ()

temp ← head

while temp ≠ null

print temp.data

temp = temp.next

end while

end function



Shot on OnePlus

chE_Reddy

```

PAGE: DATE: PAGE:
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* next;
};

struct node* head = NULL;

void createlist(int n) {
    struct node* Newnode, * temp = NULL;
    int data, i;
    if (n <= 0) {
        printf("Number of nodes should be greater than 0\n");
        return;
    }
    if (head == NULL) {
        printf("Warning: Over-writing existing list.\n");
        head = Newnode;
    }
    for (i = 1; i <= n; i++) {
        Newnode = (struct node*)malloc(sizeof(struct node));
        if (Newnode == NULL)
            printf("Memory allocation failed.\n");
        return;
    }
    printf("Enter data for node %d: ", i);
    scanf("%d", &data);
    Newnode->data = data;
    Newnode->next = NULL;
    if (head == NULL)
        head = Newnode;
    else
        temp->next = Newnode;
    chE_Reddy = Newnode;
}

```



Shot on OnePlus

chE_Reddy

DATE: _____

```

    printf("A linked list created\n");
}

void insertAtBeginning (int data) {
    struct node * Newnode = (struct node *) malloc (sizeof (struct
        node));
    if (Newnode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    Newnode-> data = data;
    Newnode-> next = head;
    head = Newnode;
    printf("Node inserted at the beginning\n");
}

void insertAtEnd (int data) { // Given a head
    struct node * Newnode = (struct node *) malloc (sizeof (struct
        node));
    if (Newnode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    Newnode-> data = data;
    Newnode-> next = NULL;
    if (head == NULL) {
        head = Newnode;
    } else {
        struct node * temp = head;
        while (temp-> next != NULL) {
            temp = temp-> next;
        }
        temp-> next = Newnode;
    }
    printf("Node inserted at the end\n");
}

```

Shot on OnePlus *inserted at the end*

chE_Reddy

DATE: _____
PAGE: _____

```

void insertatposition(int data, int pos) {
    int i;
    struct node * newnode, * temp = head;
    if (pos <= 0) {
        printf("invalid position\n");
        return;
    }
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    for (i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;
    if (temp == NULL) {
        printf("position out of range\n");
        return;
    }
    newnode = (struct node *) malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    newnode->data = data;
    newnode->next = temp->next;
    temp->next = newnode;
    printf("Node inserted at position\n");
}

void displaylist() {
    struct node * temp = head;
    if (head == NULL) {
        printf("list is empty\n");
        return;
    }
}

```



Shot on OnePlus
chE_Reddy

```

printf("Linked list:\n");
while (temp != NULL) {
    printf("%d-> ", temp->data);
    temp = temp->next;
}
printf("NULL\n");

int main() {
    int choice, n, data, pos;
    while (1) {
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at Any position\n");
        printf("4. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        if (scanf("%d", &choice) != 1) {
            while (getchar() != '\n');
            printf("Invalid input. Please enter a number\n");
            continue;
        }
        switch (choice) {
            case 1:
                createList(n);
                break;
            case 2:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                insertAtBeginning(data);
                break;
            case 3:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtPosition(pos, data);
                break;
            case 4:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                displayList();
                break;
            case 6:
                exit(0);
        }
    }
}

```



Shot on OnePlus
chE_Reddy

CASE 3:

```
printf("Enter data:");  
scanf("%d", &data);  
printf("Enter position:");  
scanf("%d", &pos);  
insertatposition(data, pos);  
break;
```

CASE 4:

```
printf("Enter data to insert");  
scanf("%d", &data);  
insertatend(data);  
break;
```

CASE 5:

```
displaylist();  
break;
```

CASE 6:

```
printf("Exiting");  
exit(0);
```

Output:

singly linked list operation.

- 1. Create linked list
- 2. Insert at Beginning
- 3. Insert at any position
- 4. Display a list
- 5. Insert at End
- 6. Exit.

Enter your choice:

Enter number of nodes: 3

Enter data for node 1: 10

Enter data for node 2: 20



Shot on OnePlus

chE_Reddy

