

ANVITA KUMAR
C-22
Roll No: 2104097

//Write a menu driven code to implement Singly Linked List

```
#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

struct node
{
    int data;

    struct node *next;
};

struct node *start = NULL;

struct node *createSLL(struct node *start);

struct node *display(struct node *start);

struct node *InsertAtBeginning(struct node *start);

struct node *InsertAtEnd(struct node *start);

struct node *InsertBefore(struct node *start);

struct node *DeleteBeginning(struct node *start);

struct node *DeleteEnd(struct node *start);

struct node *DeleteNode(struct node *start);

struct node *ForwardTraversal(struct node *start);

struct node *BackwardTraversal(struct node *start);

struct node *Sorting(struct node *start);

struct node *Count(struct node *start);

struct node *Search(struct node *start);

int main()
{
    int choice;

    start = createSLL(start);

    printf("\nSINGLY LINKED LIST CREATED\n");

    start = display(start);

    do {
```

ANVITA KUMAR

C-22

Roll No: 2104097

```
printf("\n\n****List of Operations****");

printf("\n 1: Insert at beginning");

printf("\n 2: Insert at end");

printf("\n 3: Insert at before a node");

printf("\n 4: Delete from beginning");

printf("\n 5: Delete from end");

printf("\n 6: Delete node before a specified location");

printf("\n 7: Forward Traversal");

printf("\n 8: Backward Traversal");

printf("\n 9: Sorting");

printf("\n 10: Count number of nodes");

printf("\n 11: Search an element");

printf("\n 12: EXIT");

printf("\n\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

    start = InsertAtBeginning(start);

    printf("\n");

    start = display(start);

    break;

case 2:

    start = InsertAtEnd(start);

    printf("\n");

    start = display(start);

    break;

case 3:

    start = InsertBefore(start);

    printf("\n");

    start = display(start);

    break;
```

ANVITA KUMAR
C-22
Roll No: 2104097

case 4:

```
start = DeleteBeginning(start);  
  
printf("\n");  
  
start = display(start);  
  
break;
```

case 5:

```
start = DeleteEnd(start);  
  
printf("\n");  
  
start = display(start);  
  
break;
```

case 6:

```
start = DeleteNode(start);  
  
printf("\n");  
  
start = display(start);  
  
break;
```

case 7:

```
start = ForwardTraversal(start);  
  
printf("\n");  
  
break;
```

case 8:

```
start = BackwardTraversal(start);  
  
printf("\n");  
  
start = display(start);  
  
break;
```

case 9:

```
start = Sorting(start);  
  
printf("\n");  
  
start = display(start);  
  
break;
```

case 10:

```
start = Count(start);
```

ANVITA KUMAR

C-22

Roll No: 2104097

```
        printf("\n");

        break;

    case 11:

        start = Search(start);

        printf("\n");

        break;

    case 12:

        printf("\n\tEXIT POINT");

        break;

    }

} while (choice != 12);

return 0;

}

struct node *createSLL(struct node *start)
{
    struct node *new_node, *ptr;

    int val;

    printf("\nEnter a value(enter -1 to end): ");

    scanf("%d", &val);

    while (val != -1) {

        new_node = (struct node *)malloc(sizeof(struct node));

        new_node->data = val;

        if (start == NULL) {

            new_node->next = NULL;

            start = new_node;

        }

        else {

            ptr = start;

            while (ptr->next != NULL)

                ptr = ptr->next;

            ptr->next = new_node;

        }

    }

}
```

ANVITA KUMAR

C-22

Roll No: 2104097

```
        new_node->next = NULL;

    }

    printf("Enter a value: ");

    scanf("%d", &val);

}

return start;

}

struct node *display(struct node *start)

{

    struct node *ptr;

    ptr = start;

    if (ptr == NULL) {

        printf("\tEmpty List!");

    }

    else {

        while (ptr != NULL) {

            printf("\t%d", ptr->data);

            ptr = ptr->next;

        }

    }

    return start;

}

struct node *InsertAtBeginning(struct node *start)

{

    struct node *new_node;

    int val;

    printf("Enter a value: ");

    scanf("%d", &val);

    new_node = (struct node *)malloc(sizeof(struct node));

    new_node->data = val;

    new_node->next = start;
```

ANVITA KUMAR

C-22

Roll No: 2104097

```
start = new_node;
```

```
return start;
```

```
}
```

```
struct node *InsertAtEnd(struct node *start)
```

```
{
```

```
struct node *ptr, *new_node;
```

```
int val;
```

```
printf("Enter a value: ");
```

```
scanf("%d", &val);
```

```
new_node = (struct node *)malloc(sizeof(struct node));
```

```
new_node->data = val;
```

```
new_node->next = NULL;
```

```
ptr = start;
```

```
while(ptr->next!=NULL)
```

```
ptr=ptr->next;
```

```
ptr->next=new_node;
```

```
return start;
```

```
}
```

```
struct node *InsertBefore(struct node *start)
```

```
{
```

```
struct node *new_node,*ptr,*preptr;
```

```
int val, num;
```

```
printf("Enter a value: ");
```

```
scanf("%d", &val);
```

```
printf("Enter the number before which the data has to be inserted: ");
```

```
scanf("%d", &num);
```

```
new_node = (struct node *)malloc(sizeof(struct node));
```

```
new_node->data = val;
```

```
ptr = start;
```

```
while (ptr->data != num) {
```

```
preptr = ptr;
```

ANVITA KUMAR

C-22

Roll No: 2104097

```
    ptr = ptr->next;

}

preptr -> next = new_node;

new_node -> next = ptr;

return start;

}

struct node *DeleteBeginning(struct node *start)
{
    struct node *ptr;

    ptr = start;

    start = start->next;

    free(ptr);

    return start;
}

struct node *DeleteEnd(struct node *start)
{
    struct node *ptr, *preptr;

    ptr = start;

    while (ptr->next != NULL) {

        preptr = ptr;

        ptr = ptr->next;

    }

    preptr->next = NULL;

    free(ptr);

    return start;
}

struct node *DeleteNode(struct node *start)
{
    struct node *preptr, *ptr;

    int val;

    printf("Enter the value before which the data has to be deleted: ");
```

ANVITA KUMAR

C-22

Roll No: 2104097

```
scanf("%d", &val);
```

```
ptr = start;
```

```
if(ptr->data == val-1) {
```

```
    start = DeleteBeginning(start);
```

```
    return start;
```

```
}
```

```
else {
```

```
    while(ptr->data != val-1) {
```

```
        preptr = ptr;
```

```
        ptr = ptr->next;
```

```
    }
```

```
    preptr->next = ptr->next;
```

```
    free(ptr);
```

```
    return start;
```

```
}
```

```
}
```

```
struct node *ForwardTraversal(struct node *start)
```

```
{
```

```
    struct node *ptr;
```

```
    ptr = start;
```

```
    if (ptr == NULL) {
```

```
        printf("\tEmpty List!");
```

```
    }
```

```
    else {
```

```
        printf("\n");
```

```
        while (ptr != NULL) {
```

```
            printf("\t%d", ptr->data);
```

```
            ptr = ptr->next;
```

```
        }
```

```
    }
```

```
    return start;
```


ANVITA KUMAR
C-22
Roll No: 2104097
}

struct node *BackwardTraversal(struct node *start)

```
{  
    struct node* prev = NULL;  
    struct node* current = start;  
    struct node* next = NULL;  
    while (current != NULL) {  
        next = current->next;  
        current->next = prev;  
        prev = current;  
        current = next;  
    }  
    start = prev;  
}
```

struct node *Sorting(struct node *start)

```
{  
    struct node *ptr1, *ptr2;  
    int temp;  
    ptr1 = start;  
    while (ptr1->next != NULL) {  
        ptr2 = ptr1->next;  
        while (ptr2 != NULL) {  
            if (ptr1->data > ptr2->data) {  
                temp = ptr1->data;  
                ptr1->data = ptr2->data;  
                ptr2->data = temp;  
            }  
            ptr2 = ptr2->next;  
        }  
        ptr1 = ptr1->next;  
    }  
}
```

ANVITA KUMAR
C-22
Roll No: 2104097

```
    return start;
}

struct node *Count(struct node *start)
{
    int i;
    i=0;
    while(start!=NULL) {
        i=i+1;
        start=start->next;
    }
    printf("Number of nodes in the list: %d", i);
}

struct node *Search(struct node *start)
{
    struct node* current;
    int val;
    printf("Enter a value that is to be searched: ");
    scanf("%d", &val);
    if(start == NULL)    printf("\tEmpty List!");
    else {
        current = start;
        while (current != NULL) {
            if (current -> data == val)    printf("\tElement found");
            break;
        }
        current = current->next;
    }
    if(current == NULL) {
        printf("\tElement not found");
    }
}
```

ANVITA KUMAR
C-22
Roll No: 2104097

```
exp6.c - exp2 - Visual Studio Code
C exp6.c U X
.vscode> C exp6.c
1 #include<stdio.h>

PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER

> TERMINAL
PS C:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2> cd "c:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode\" ; if ($?) { gcc exp6.c -o exp6 } ; if ($?) { .\exp6 }

Enter a value(enter -1 to end): 2
Enter a value: 3
Enter a value: 4
Enter a value: -1

SINGLY LINKED LIST CREATED
2 3 4

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 1
Enter a value: 1

1 2 3 4

****List of Operations****
```

```
exp6.c - exp2 - Visual Studio Code
C exp6.c U X
.vscode> C exp6.c
1 #include<stdio.h>

PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER

> TERMINAL
Enter your choice: 2
Enter a value: 6

1 2 3 4 6

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 3
Enter a value: 5
Enter the number before which the data has to be inserted: 6

1 2 3 4 5 6

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
```

ANVITA KUMAR
C-22
Roll No: 2104097

```
exp6.c exp2 - Visual Studio Code
C exp6.c U X
.vscode > C exp6.c
1 #include<stdio.h>

PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER

> TERMINAL
Enter your choice: 4

      2      3      4      5      6

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 5

      2      3      4      5

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
```

```
exp6.c exp2 - Visual Studio Code
C exp6.c U X
.vscode > C exp6.c
1 #include<stdio.h>
2 struct node
3 {
4     int data;
5     struct node *next;
6 };
7 struct node *head = NULL;
8 struct node *tail = NULL;
9
10 void insert_at_beginning(int data)
11 {
12     struct node *new_node = (struct node *)malloc(sizeof(struct node));
13     new_node->data = data;
14     new_node->next = head;
15     head = new_node;
16 }
17 void insert_at_end(int data)
18 {
19     struct node *new_node = (struct node *)malloc(sizeof(struct node));
20     new_node->data = data;
21     new_node->next = NULL;
22     if (tail == NULL)
23         tail = new_node;
24     else
25         tail->next = new_node;
26     tail = new_node;
27 }
28 void insert_at_before_a_node(int data, int before)
29 {
30     struct node *new_node = (struct node *)malloc(sizeof(struct node));
31     new_node->data = data;
32     new_node->next = NULL;
33     if (head == NULL)
34         head = new_node;
35     else
36     {
37         struct node *temp = head;
38         while (temp->next != NULL)
39             temp = temp->next;
40         temp->next = new_node;
41     }
42 }
43 void delete_from_beginning()
44 {
45     if (head == NULL)
46         return;
47     head = head->next;
48 }
49 void delete_from_end()
50 {
51     if (tail == NULL)
52         return;
53     if (head == tail)
54         head = NULL;
55     else
56     {
57         struct node *temp = head;
58         while (temp->next != tail)
59             temp = temp->next;
60         temp->next = NULL;
61     }
62 }
63 void delete_node_before_a_specified_location(int before)
64 {
65     if (head == NULL)
66         return;
67     if (head->next == NULL)
68         return;
69     struct node *temp = head;
70     while (temp->next != NULL)
71     {
72         if (temp->next->data == before)
73         {
74             temp->next = temp->next->next;
75             return;
76         }
77         temp = temp->next;
78     }
79 }
80 void forward_traversal()
81 {
82     struct node *temp = head;
83     while (temp != NULL)
84     {
85         printf("%d ", temp->data);
86         temp = temp->next;
87     }
88     printf("\n");
89 }
90 void backward_traversal()
91 {
92     struct node *temp = tail;
93     while (temp != NULL)
94     {
95         printf("%d ", temp->data);
96         temp = temp->next;
97     }
98     printf("\n");
99 }
100 void sorting()
101 {
102     struct node *temp = head;
103     struct node *new_node = NULL;
104     while (temp != NULL)
105     {
106         new_node = (struct node *)malloc(sizeof(struct node));
107         new_node->data = temp->data;
108         new_node->next = NULL;
109         if (head == NULL)
110             head = new_node;
111         else
112             head->next = new_node;
113         temp = temp->next;
114     }
115 }
116 void count_number_of_nodes()
117 {
118     struct node *temp = head;
119     int count = 0;
120     while (temp != NULL)
121     {
122         count++;
123         temp = temp->next;
124     }
125     printf("Count number of nodes: %d\n", count);
126 }
127 void search_an_element(int data)
128 {
129     struct node *temp = head;
130     while (temp != NULL)
131     {
132         if (temp->data == data)
133             printf("Element found at %d\n", temp->data);
134         temp = temp->next;
135     }
136 }
137 void EXIT()
138 {
139     printf("EXIT\n");
140 }
141 int main()
142 {
143     int choice;
144     while (1)
145     {
146         printf("Enter your choice: ");
147         scanf("%d", &choice);
148         switch (choice)
149         {
150             case 1: insert_at_beginning(10); break;
151             case 2: insert_at_end(20); break;
152             case 3: insert_at_before_a_node(30, 10); break;
153             case 4: delete_from_beginning(); break;
154             case 5: delete_from_end(); break;
155             case 6: delete_node_before_a_specified_location(10); break;
156             case 7: forward_traversal(); break;
157             case 8: backward_traversal(); break;
158             case 9: sorting(); break;
159             case 10: count_number_of_nodes(); break;
160             case 11: search_an_element(10); break;
161             case 12: EXIT(); break;
162             default: printf("Invalid choice\n");
163         }
164     }
165 }
```

ANVITA KUMAR
C-22
Roll No: 2104097

```
File Edit Selection View Go Run Terminal Help
exp6.c - exp2 - Visual Studio Code

C exp6.c U X
.vscode > C exp6.c

PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER

TERMINAL
Enter your choice: 8

5 3 2

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 9

2 3 5

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
```

```
File Edit Selection View Go Run Terminal Help
exp6.c - exp2 - Visual Studio Code

C exp6.c U X
.vscode > C exp6.c
1 #include<stdio.h>

PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER

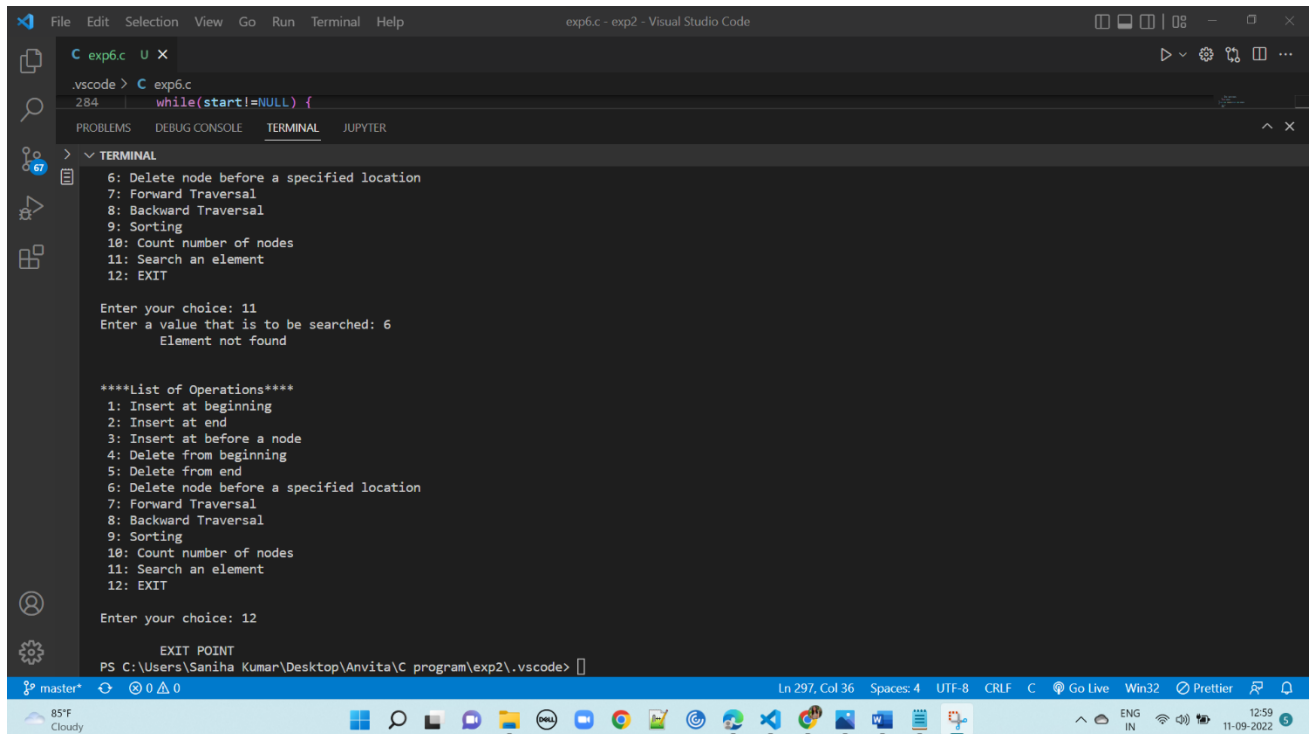
TERMINAL
Enter your choice: 10
Number of nodes in the list: 3

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 11
Enter a value that is to be searched: 3
Element found

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
```

ANVITA KUMAR
C-22
Roll No: 2104097



The image shows a Visual Studio Code editor window with a C program named `exp6.c` and its terminal output. The program implements a linked list with various operations. The terminal shows the user selecting option 11 (Search an element) and entering the value 6, which results in "Element not found". It then shows the user selecting option 12 (EXIT).

```
exp6.c - exp2 - Visual Studio Code
C exp6.c U X
.vscode > C exp6.c
284 while(start!=NULL) {
PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER
TERMINAL
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 11
Enter a value that is to be searched: 6
Element not found

****List of Operations****
1: Insert at beginning
2: Insert at end
3: Insert at before a node
4: Delete from beginning
5: Delete from end
6: Delete node before a specified location
7: Forward Traversal
8: Backward Traversal
9: Sorting
10: Count number of nodes
11: Search an element
12: EXIT

Enter your choice: 12

EXIT POINT
PS C:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode>
```