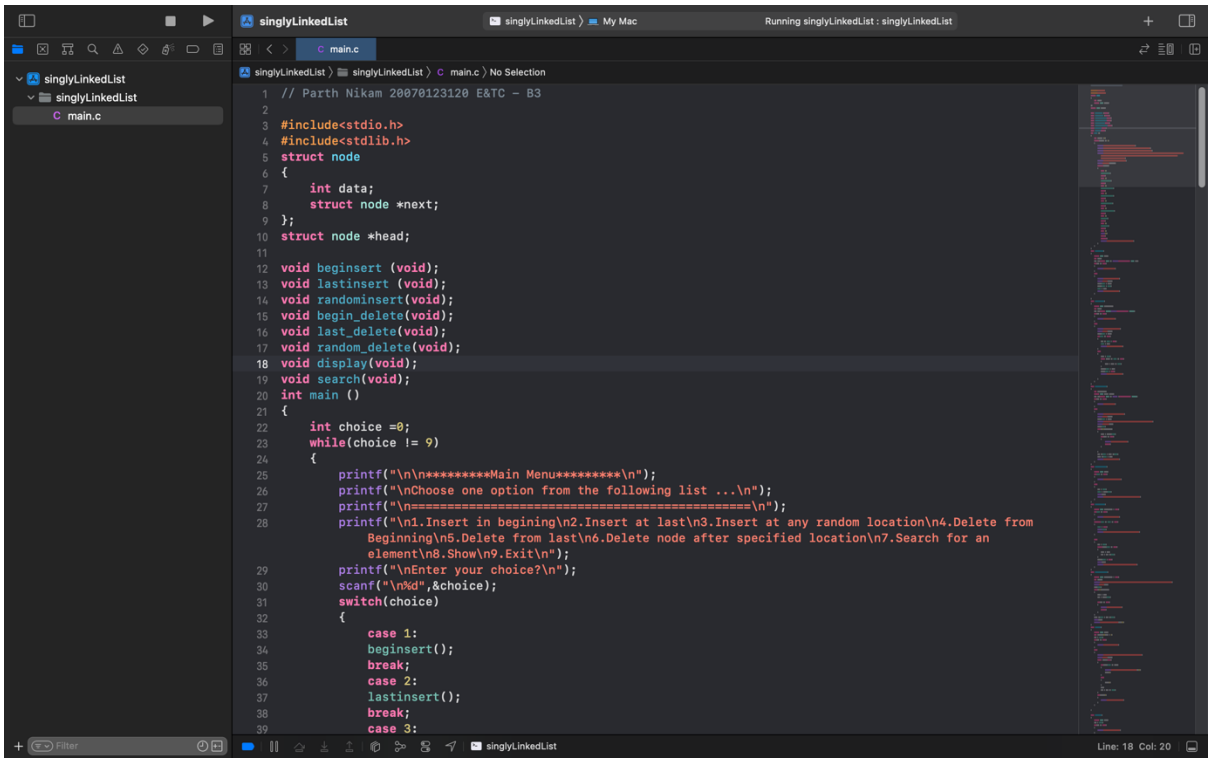
Data Structures using C

Parth Nikam
20070123120
E&TC – B3

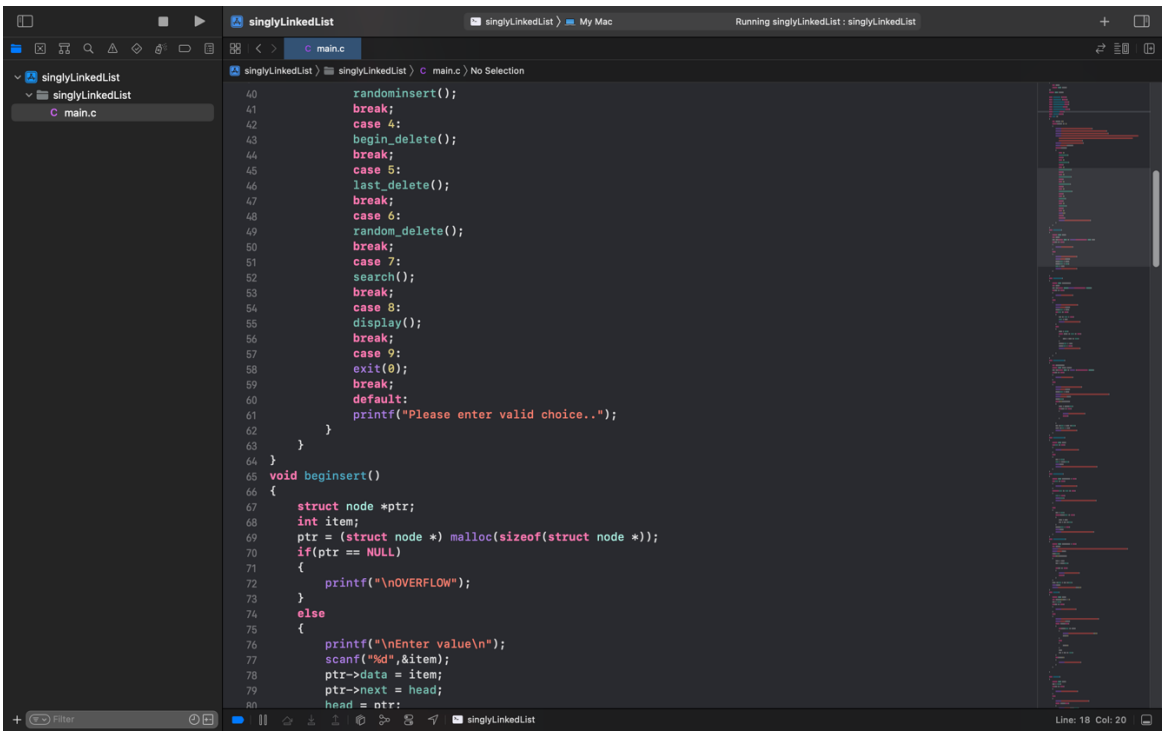
Aim: - Studying and coding on Singly Linked List.

Objective: - To perform Singly Linked List in C language

Code: -



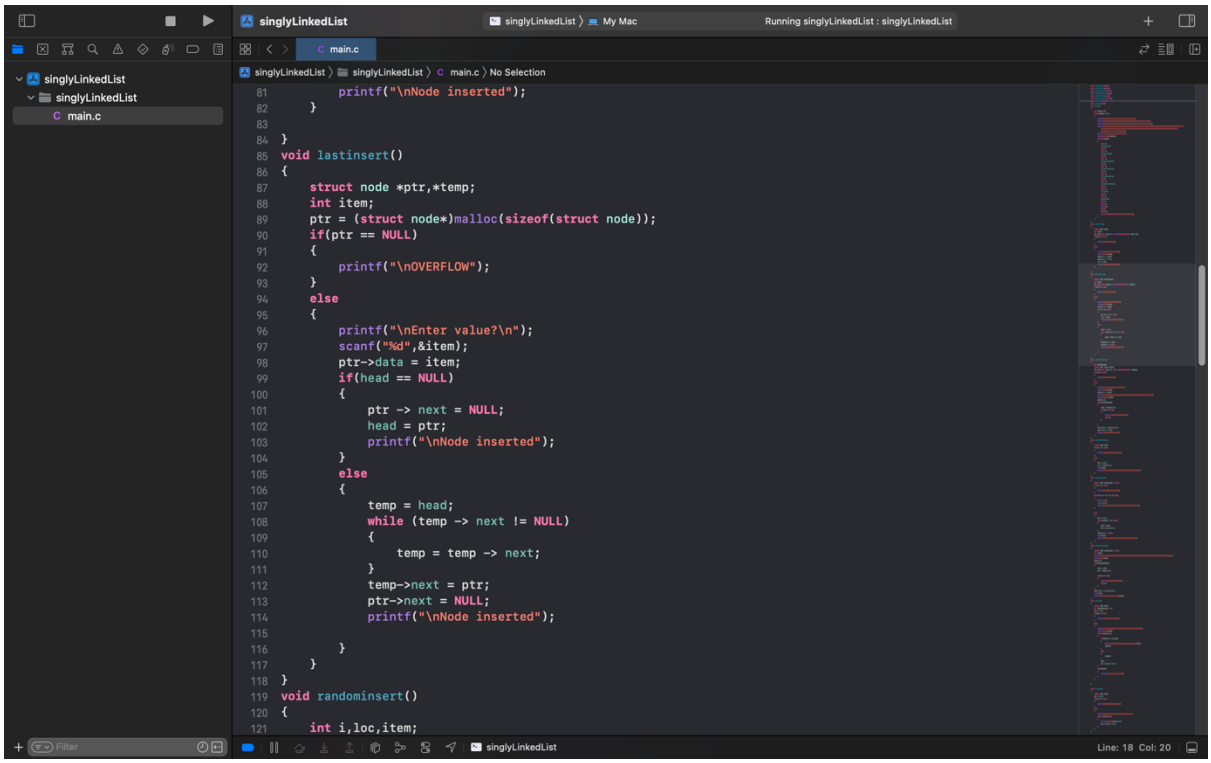
```c
// Parth Nikam 20070123120 E&TC – B3

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

void beginsert (void);
void lastinsert (void);
void randominsert(void);
void begin_delete(void);
void last_delete(void);
void random_delete(void);
void display(void);
void search(void);
int main ()
{
    int choice =0;
    while(choice != 9)
    {
        printf("\n\n*********Main Menu*********\n");
        printf("\nChoose one option from the following list ...\n");
        printf("\n===============================================\n");
        printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from
            Beginning\n5.Delete from last\n6.Delete node after specified location\n7.Search for an
            element\n8.Show\n9.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            beginsert();
            break;
            case 2:
            lastinsert();
            break;
            case 3:
```



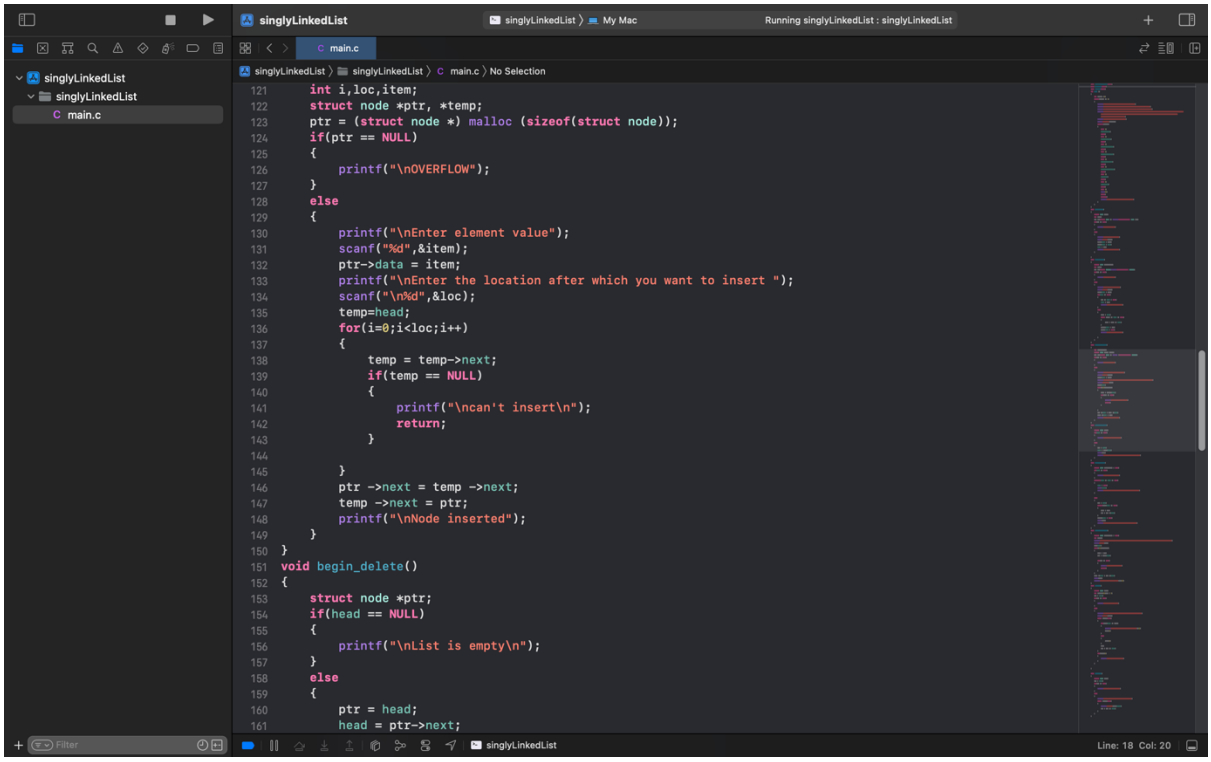
```c
            randominsert();
            break;
            case 4:
            begin_delete();
            break;
            case 5:
            last_delete();
            break;
            case 6:
            random_delete();
            break;
            case 7:
            search();
            break;
            case 8:
            display();
            break;
            case 9:
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void beginsert()
{
    struct node *ptr;
    int item;
    ptr = (struct node *) malloc(sizeof(struct node *));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter value\n");
        scanf("%d",&item);
        ptr->data = item;
        ptr->next = head;
        head = ptr;
```

Data Structures using C

Parth Nikam
20070123120
E&TC – B3



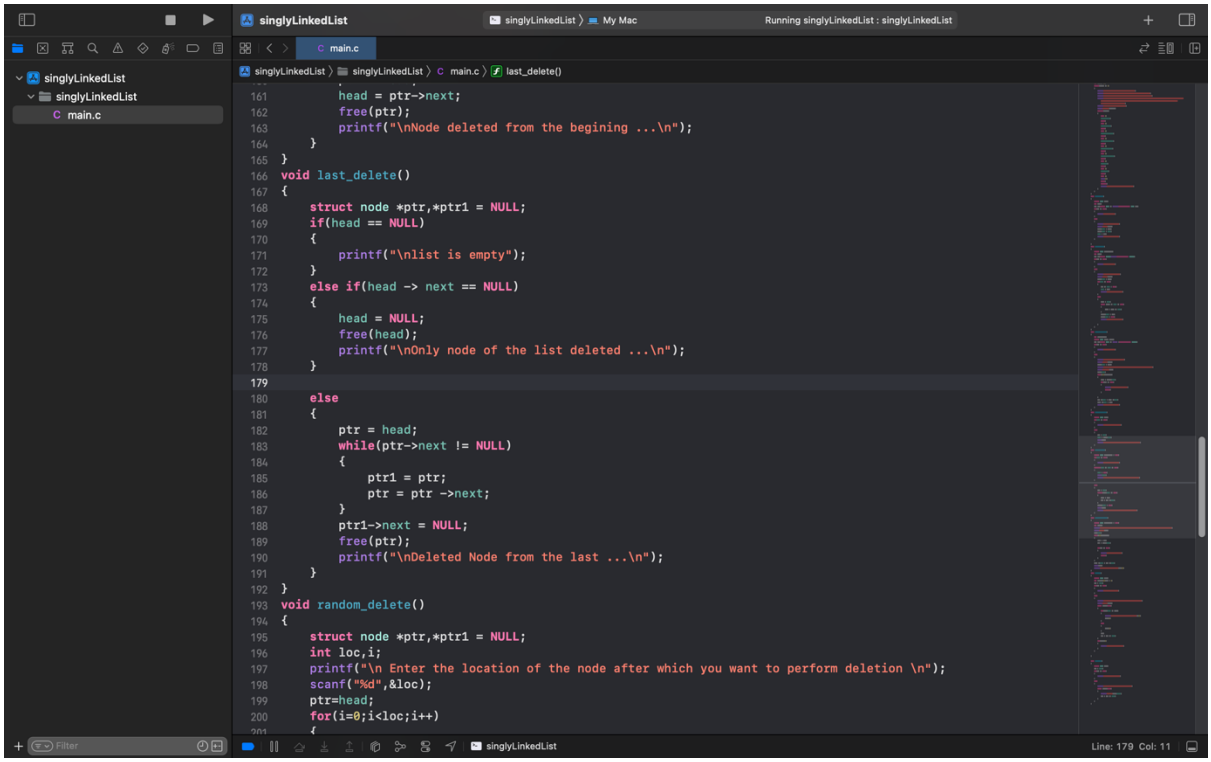
```
        printf("\nNode inserted");
    }

}
void lastinsert()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter value?\n");
        scanf("%d",&item);
        ptr->data = item;
        if(head == NULL)
        {
            ptr -> next = NULL;
            head = ptr;
            printf("\nNode inserted");
        }
        else
        {
            temp = head;
            while (temp -> next != NULL)
            {
                temp = temp -> next;
            }
            temp->next = ptr;
            ptr->next = NULL;
            printf("\nNode inserted");

        }
    }
}
void randominsert()
{
    int i,loc,item;
```



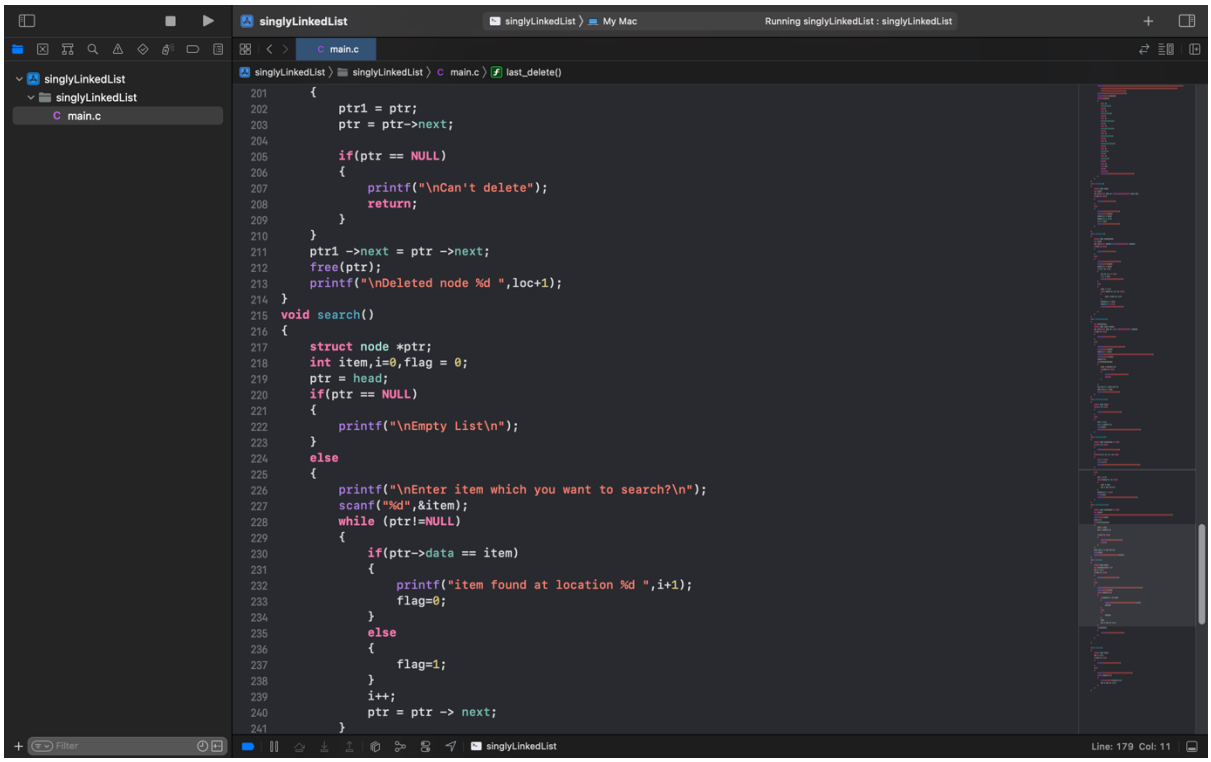
```
    int i,loc,item;
    struct node *ptr, *temp;
    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter element value");
        scanf("%d",&item);
        ptr->data = item;
        printf("\nEnter the location after which you want to insert ");
        scanf("\n%d",&loc);
        temp=head;
        for(i=0;i<loc;i++)
        {
            temp = temp->next;
            if(temp == NULL)
            {
                printf("\ncan't insert\n");
                return;
            }

        }
        ptr ->next = temp ->next;
        temp ->next = ptr;
        printf("\nNode inserted");
    }
}
void begin_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nList is empty\n");
    }
    else
    {
        ptr = head;
        head = ptr->next;
```

Data Structures using C

Parth Nikam
20070123120
E&TC – B3



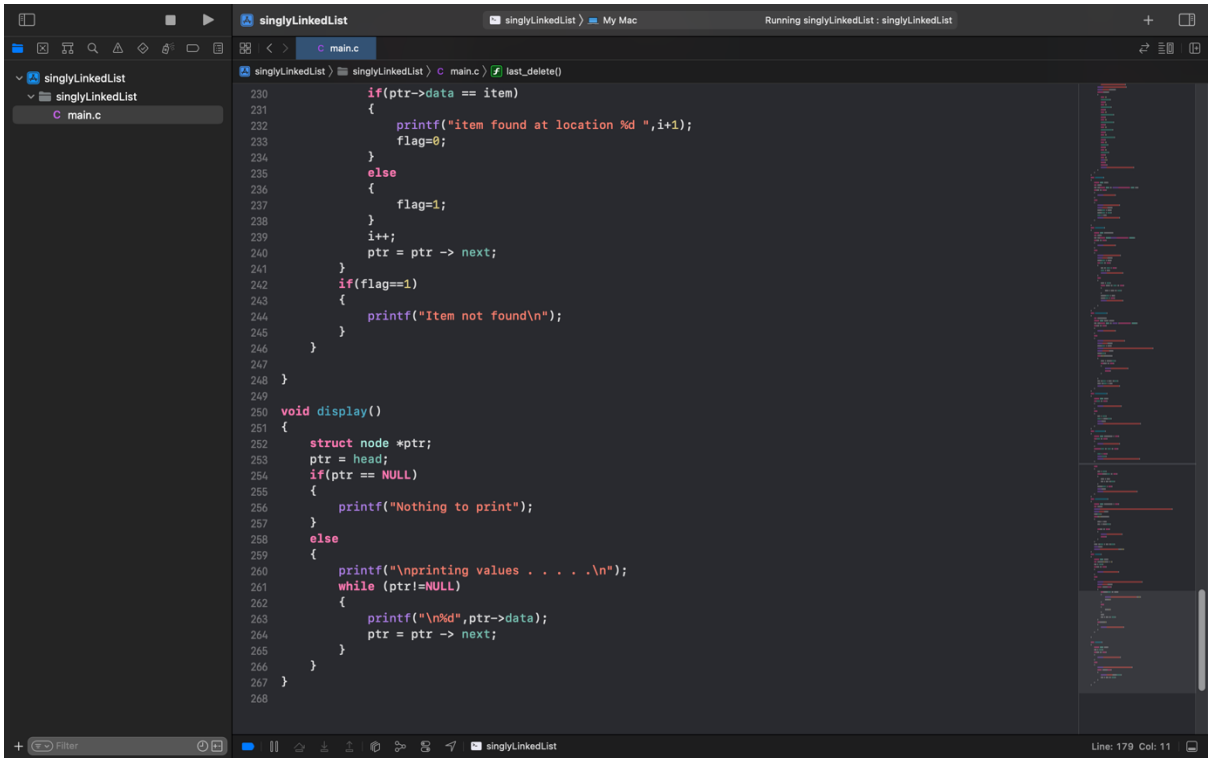
```
161         head = ptr->next;
162         free(ptr);
163         printf("\nNode deleted from the begining ...\n");
164     }
165 }
166 void last_delete()
167 {
168     struct node *ptr,*ptr1 = NULL;
169     if(head == NULL)
170     {
171         printf("\nlist is empty");
172     }
173     else if(head -> next == NULL)
174     {
175         head = NULL;
176         free(head);
177         printf("\nOnly node of the list deleted ...\n");
178     }
179
180     else
181     {
182         ptr = head;
183         while(ptr->next != NULL)
184         {
185             ptr1 = ptr;
186             ptr = ptr ->next;
187         }
188         ptr1->next = NULL;
189         free(ptr);
190         printf("\nDeleted Node from the last ...\n");
191     }
192 }
193 void random_delete()
194 {
195     struct node *ptr,*ptr1 = NULL;
196     int loc,i;
197     printf("\n Enter the location of the node after which you want to perform deletion \n");
198     scanf("%d",&loc);
199     ptr=head;
200     for(i=0;i<loc;i++)
```



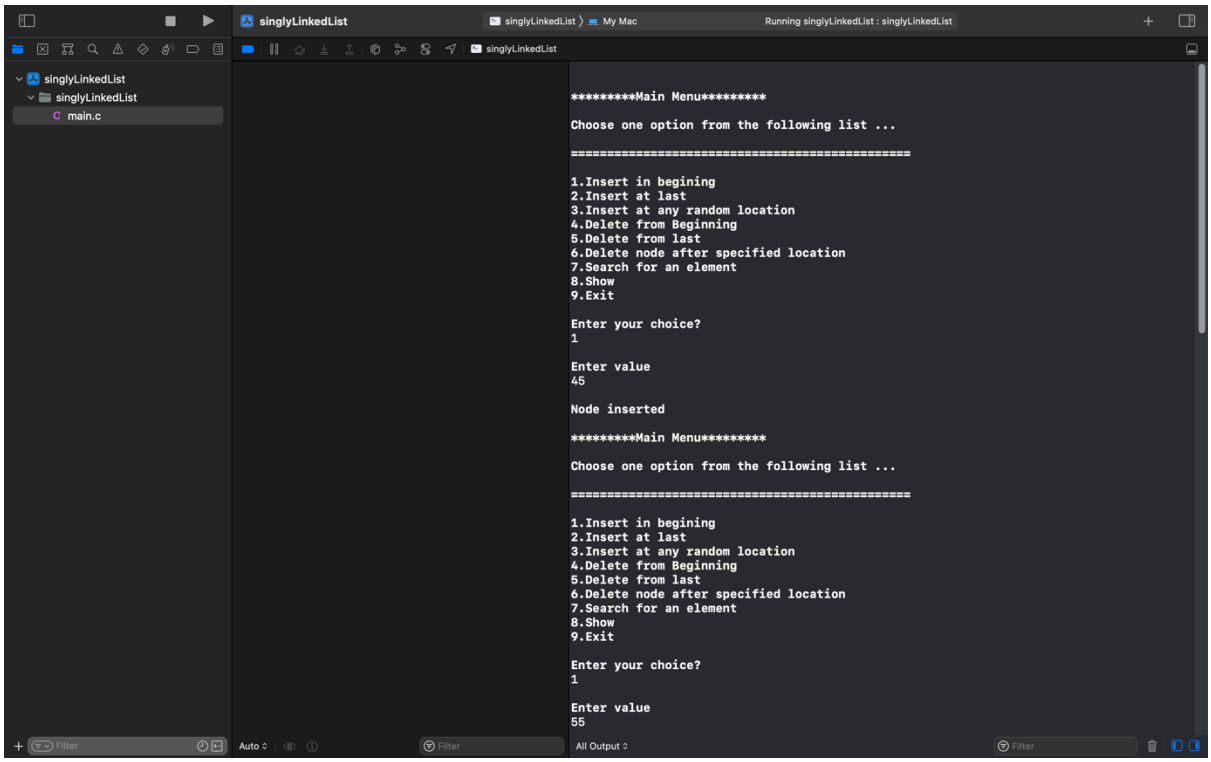
```
201     {
202         ptr1 = ptr;
203         ptr = ptr->next;
204
205         if(ptr == NULL)
206         {
207             printf("\nCan't delete");
208             return;
209         }
210     }
211     ptr1 ->next = ptr ->next;
212     free(ptr);
213     printf("\nDeleted node %d ",loc+1);
214 }
215 void search()
216 {
217     struct node *ptr;
218     int item,i=0,flag = 0;
219     ptr = head;
220     if(ptr == NULL)
221     {
222         printf("\nEmpty List\n");
223     }
224     else
225     {
226         printf("\nEnter item which you want to search?\n");
227         scanf("%d",&item);
228         while (ptr!=NULL)
229         {
230             if(ptr->data == item)
231             {
232                 printf("item found at location %d ",i+1);
233                 flag=0;
234             }
235             else
236             {
237                 flag=1;
238             }
239             i++;
240             ptr = ptr -> next;
241         }
```

Data Structures using C

Parth Nikam
20070123120
E&TC – B3



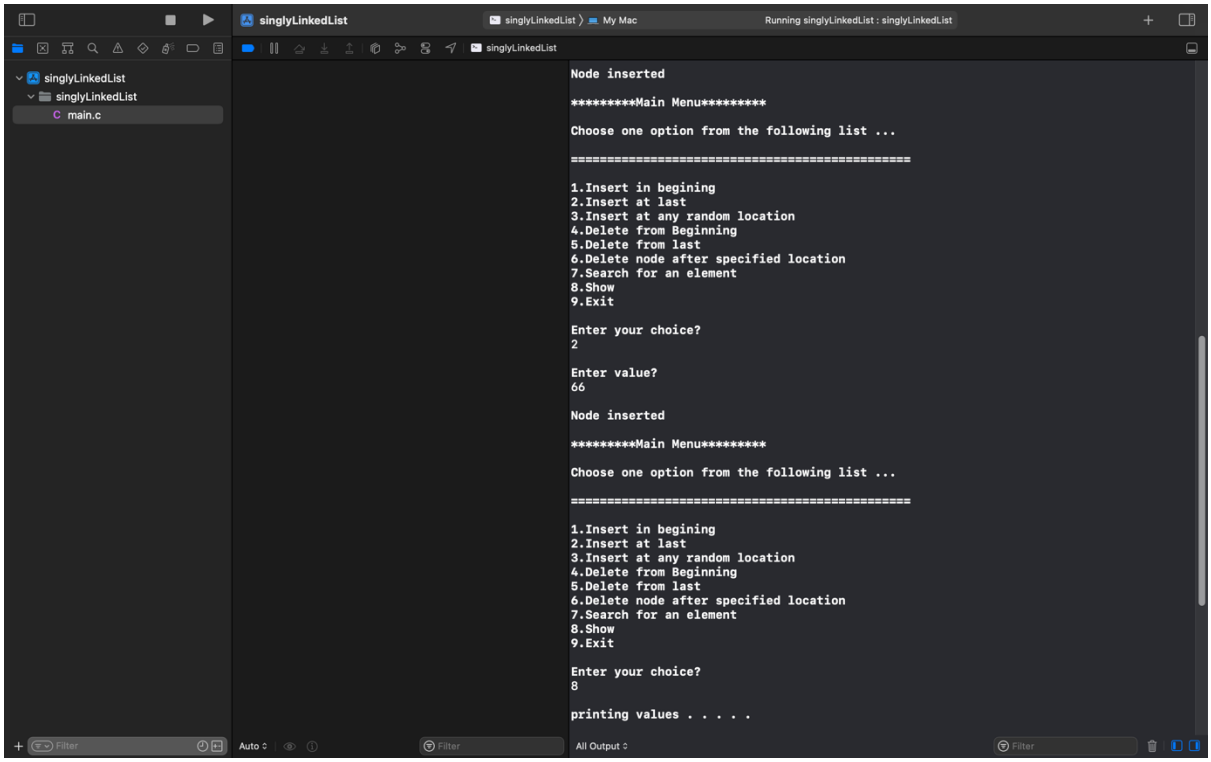
```c
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
        if(flag==1)
        {
            printf("Item not found\n");
        }
    }

}

void display()
{
    struct node *ptr;
    ptr = head;
    if(ptr == NULL)
    {
        printf("Nothing to print");
    }
    else
    {
        printf("\nprinting values . . . . .\n");
        while (ptr!=NULL)
        {
            printf("\n%d",ptr->data);
            ptr = ptr -> next;
        }
    }
}
```

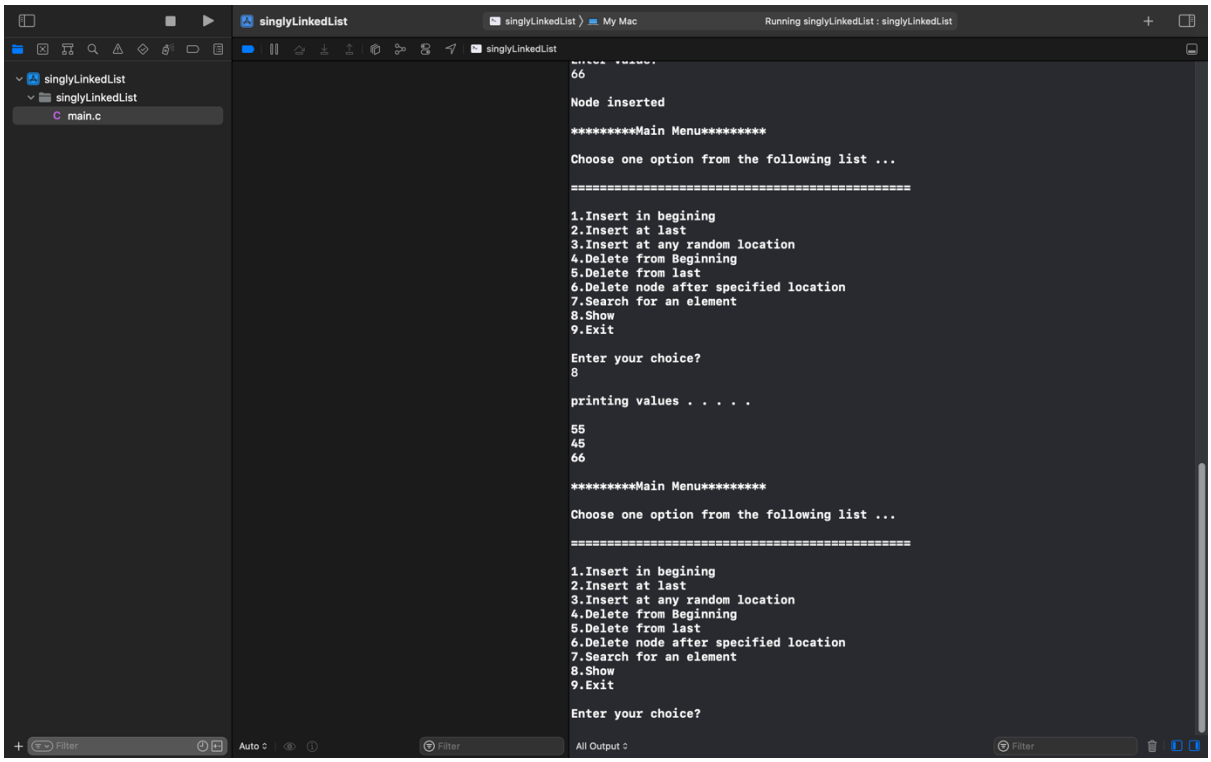


```
*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Search for an element
8.Show
9.Exit

Enter your choice?
1

Enter value
45

Node inserted

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Search for an element
8.Show
9.Exit

Enter your choice?
1

Enter value
55
```

Data Structures using C

Parth Nikam
20070123120
E&TC – B3





Result: - Singly Linked List is performed and studies successfully.