Date: 19.09.2023 Name: Niranjan.B
Ex. No: 3 Reg. No.: 3122225001082

UCS 2312 Data Structures Lab

Assignment 3: Doubly Linked List and its applications

Date of Assignment: 19.09.2023

Create an ADT for the doubly linked list data structure with the following functions. Each node which consists of integer data, address of left and right nodes [CO1, K3]

Create a ListADT which has implementations for the following operations

1. Insert an item in the front of the list
   void insertFront(listADT L, int c)
2. Insert an item at the end of the list
   void insertEnd(listADT L, int c)
3. Insert an item 'd' after the first occurrence 'c' of the list
   void insertMiddle(listADT L, int c, int d)
4. Display the items from the list
   void displayItems(listADT L)
5. Delete the item present in the list
   void deleteItem(listADT L, int c)
6. Search an element in the list and return the number of occurrences
   int searchItem(listADT L, int c)

Write a program in C to test the ListADT for its operations with the following test cases.

Testcase:

Initially L is Empty

insertFront(L,6) → header←→6

insertEnd(L,2) → header←→2←→6

insertMiddle(L,2,1) → header←→2←→1←→6

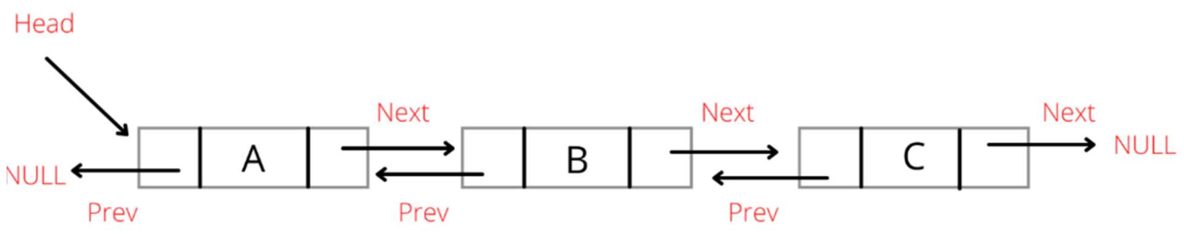
insertMiddle(L,2,1) → header←→2←→1←→1←→6

search(L,1) → 2

In addition, do the following operations:

1. Check whether the list contains duplicates?
2. Create separate lists containing even and odd numbers from the list
3. Add two 10-digit numbers using the list

Data Structure – Double Linked List:



Algorithm –

Algorithm: Checks whether the list contains duplicates

Input – Pointer to header node

Output – int

```
1.  ptr1= header->right
2.  while (ptr1 != NULL)
        ptr2 = ptr1->right
        while (ptr2 != NULL)
            if (ptr1->data == ptr2->data)
                return 1
            ptr2 = ptr2->right
        ptr1 = ptr1->right
3.  return 0
```

Algorithm: Create separate lists containing even and odd numbers from the list

Input – Pointer to header, pointer to odd header, pointer to even header

Output – void

```
1.  ptr = header->right
2.  while (ptr != NULL)
        if (ptr->data % 2 == 0)
            insertEnd(even_head, ptr->data)
        else
            insertEnd(odd_head, ptr->data)
        ptr = ptr->right
```

Algorithm: Add two 10-digit numbers using the list

Input – Pointer to number1 header, pointer to number2 header

Output – struct node *

```
1.  p1 = n1->right
2.  p2 = n2->right
3.  res->left = NULL
4.  res->right = NULL
5.  carry = 0
6.  while (p1 != NULL)
        end1 = p1
        end2 = p2
        p1 = p1->right
        p2 = p2->right
7.  p1 = end1
8.  p2 = end2
9.  while (p1 != n1)
        sum = p1->data + p2->data + carry
        if sum > 9
            carry = 1
            insertFront(res, sum%10)
        else
            carry = 0
            insertFront(res, sum)
        p1= p1->left
        p2 = p2->left
10. if carry==1
        insertFront(res, 1)
11. return res
```

DLinkedListADT.h code:

```
struct node
{
  int data;
  struct node* left;
  struct node* right;
};

void insertFront(struct node* header, int c)
{
  struct node* temp;
  temp=(struct node*)malloc(sizeof(struct node));
  temp->data=c;
  if(header->right==NULL)
  {
    temp->right=header->right;
    temp->left=header;
    header->right=temp;
  }
  else
  {
    struct node* ptr;
    ptr=header->right;
    temp->right=ptr;
    ptr->left=temp;
    header->right=temp;
    temp->left=header;
  }
}

void displayItems(struct node* header)
{
  struct node* ptr, *end;
  ptr=header->right;
  printf("\nForward: ");
  while(ptr!=NULL)
    {
      printf("%d ", ptr->data);
      end=ptr;
      ptr=ptr->right;
    }
  printf("Backward: ");
  while(end!=header)
    {
      printf("%d ", end->data);
      end=end->left;
    }
}

struct node* search(struct node* header, int key)
{
  struct node* ptr;
  ptr=header->right;
  while(ptr!=NULL)
    {
      if(ptr->data==key)
        return ptr;
      ptr=ptr->right;
    }
  return NULL;
}

void insertMiddle(struct node* header, int key, int data)
{
  struct node* temp;
  temp=(struct node*)malloc(sizeof(struct node));
  struct node *ptr, *next;
  ptr=search(header,key);
  if(ptr==NULL)
    printf("\nNot found.");
  else
  {
    temp->data=data;
    next=ptr->right;
    ptr->right=temp;
    temp->left=ptr;
    temp->right=next;
    next->left=temp;
  }
}

void insertEnd(struct node* header, int data)
{
  struct node* temp;
  temp=(struct node*)malloc(sizeof(struct node));
  struct node *ptr;
  ptr=header;
  while(ptr->right!=NULL)
      ptr=ptr->right;
  temp->data=data;
  ptr->right=temp;
  temp->left=ptr;
  temp->right=NULL;
}

void deleteItem(struct node* header, int data)
{
  struct node *prev, *next, *ptr;
  ptr=search(header,data);
  prev=ptr->left;
  next=ptr->right;
  prev->right=next;
  next->left=prev;
  free(ptr);
}

int searchItem(struct node* header, int c)
{
  int count=0;
  struct node* ptr=header->right;
  while(ptr!=NULL)
    {
      if(ptr->data==c)
        ++count;
      ptr=ptr->right;
    }
  return count;
}

int duplicates(struct node* header)
{
  int flag;
  struct node* ptr1=header->right, *ptr2;
  while(ptr1!=NULL)
    {
      ptr2=ptr1->right;
      while(ptr2!=NULL)
        {
          if(ptr1->data==ptr2->data)
            return 1;
          ptr2=ptr2->right;
        }
      ptr1=ptr1->right;
    }
  return 0;
}


void evenOdd(struct node* header,struct node* even_head,struct node*
odd_head)
{
  struct node *ptr = header->right;
  while (ptr != NULL)
  {
    if ((ptr->data) % 2 == 0)
      insertEnd(even_head, ptr->data);
    else
      insertEnd(odd_head, ptr->data);
    ptr = ptr->right;
  }
}


int palindrome(struct node* header)
{
  struct node* ptr=header->right;
  struct node* end;
  while(ptr!=NULL)
  {
    end=ptr;
    ptr=ptr->right;
  }
  if(header->right!=NULL)
  {
    ptr=header->right;
    while(ptr!=NULL && end!=header)
    {
      if(ptr->data!=end->data)
        return 0;
      ptr=ptr->right;
      end=end->left;
      if(ptr==end)
        break;
    }
  }
  return 1;
}

struct node* add10Digit(struct node* n1,struct node* n2)
{
  struct node* res=(struct node*)malloc(sizeof(struct node));
  struct node *p1,*p2,*end1,*end2;
  p1=n1->right;
  p2=n2->right;
  res->left=NULL;
  res->right=NULL;
  int sum,carry=0;
  while(p1!=NULL)
  {
    end1=p1;
    end2=p2;
    p1=p1->right;
    p2=p2->right;
  }
  p1=end1;
  p2=end2;
  while(p1!=n1)
  {
    sum=p1->data+p2->data+carry;
    if(sum>9)
    {
      carry=1;
      insertFront(res,sum%10);
    }
    else
    {
      carry=0;
      insertFront(res,sum);
    }
    p1=p1->left;
    p2=p2->left;
  }
  if(carry==1)
  {
    insertFront(res,1);
  }
  return res;
}
```

Main.c code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "DLinkedListADT.h"

void main ()
{
  struct node* header;
  header = (struct node*)malloc(sizeof(struct node));
  header->left = NULL;
  header->right = NULL;
  int choice;
  while(choice + 1)
      {
```

```
		printf("\n-1: EXIT\n 0: DISPLAY ITEMS\n 1: INSERT AT FRONT\n 2: INSERT AT END\n 3: INSERT AT MIDDLE\n 4: DELETE ITEM");
		printf("\n 5: SEARCH ITEM\n 6: CHECK DUPLICATES\n 7: SEPARATE EVEN AND ODD\n 8: ADD 10 DIGIT NO.\n 9: CHECK PALINDROME\nChoice : ");
		scanf("%d", &choice);
		switch (choice)
		{
			case -1: break;
			case 0:
			{
				displayItems(header);
					printf("\n");
				break;
			}
			case 1:
			{
				printf("\nENTER NEW ELEMENT: ");
				int data;
				scanf("%d", &data);
				insertFront(header,data);
				displayItems(header);
					printf("\n");
				break;
			}
			case 2:
			{
				printf("\nENTER NEW ELEMENT: ");
				int data;
				scanf("%d", &data);
				insertEnd(header,data);
				displayItems(header);
					printf("\n");
				break;
			}
```

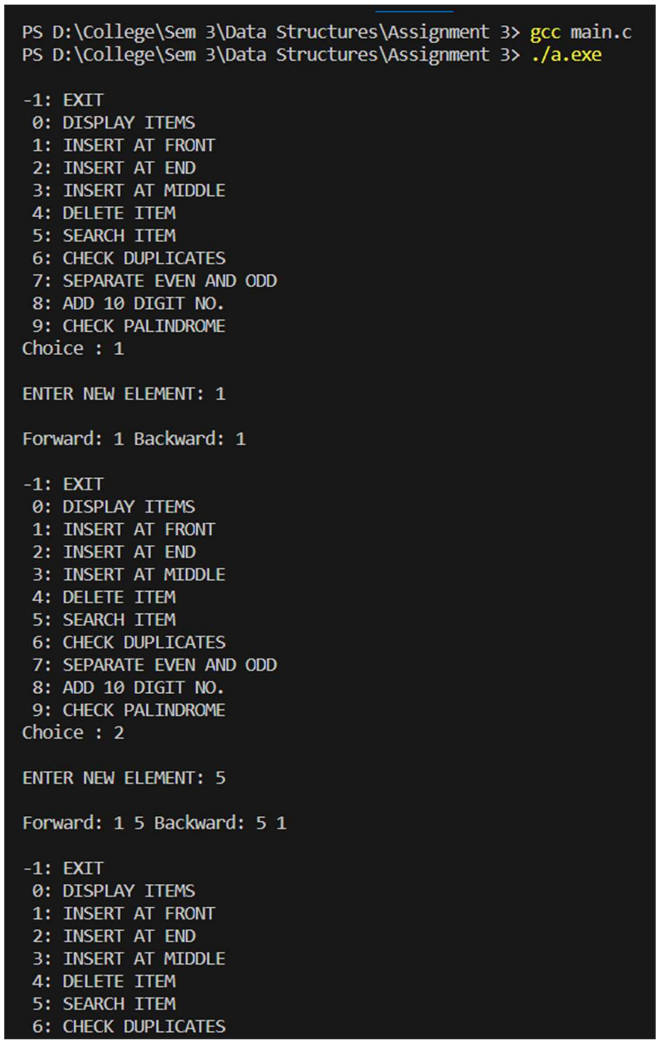
```
                        case 3:
                {
                        printf("\nENTER NEW ELEMENT AND TO INSERT AFTER:
");
                        int data,key;
                        scanf("%d", &data);
                            scanf("%d", &key);
                        insertMiddle(header,key,data);
                        displayItems(header);
                            printf("\n");
                        break;
                }
                        case 4:
                {
                        printf("\nENTER ELEMENT: ");
                        int data;
                        scanf("%d", &data);
                        deleteItem(header,data);
                            printf("ELEMENT DELETED\n");
                        displayItems(header);
                            printf("\n");
                        break;
                }
                        case 5:
                {
                        int search,c;
                            printf("\nEnter element to search: ");
                            scanf("%d", &search);
                            c=searchItem(header,search);
                            printf("\nThe no. of occurences of %d in the
list is %d.\n",search, c);
                        break;
                }
                        case 6:
```

```
                    {
                                if(!duplicates(header))
                                     printf("\nNo duplicates present.");
                                else
                                     printf("\nDuplicates present.");
                                printf("\n");
                          break;
                    }
                          case 7:
                    {
                                struct node *even_head = (struct node
*)malloc(sizeof(struct node));
                                even_head->left = NULL;
                                even_head->right = NULL;
                                struct node *odd_head = (struct node
*)malloc(sizeof(struct node));
                                odd_head->left = NULL;
                                odd_head->right = NULL;
                          evenOdd(header,even_head,odd_head);
                                printf("\nEven List: ");
                                displayItems(even_head);
                                printf("\nOdd List: ");
                                displayItems(odd_head);
                                printf("\n");
                          break;
                    }
                    case 8:
                    {
                          struct node *n1=(struct node *)malloc(sizeof(struct
node));
                          n1->left=NULL;
                          n1->right=NULL;
                          struct node *n2=(struct node *)malloc(sizeof(struct
node));
                          n2->left=NULL;
```

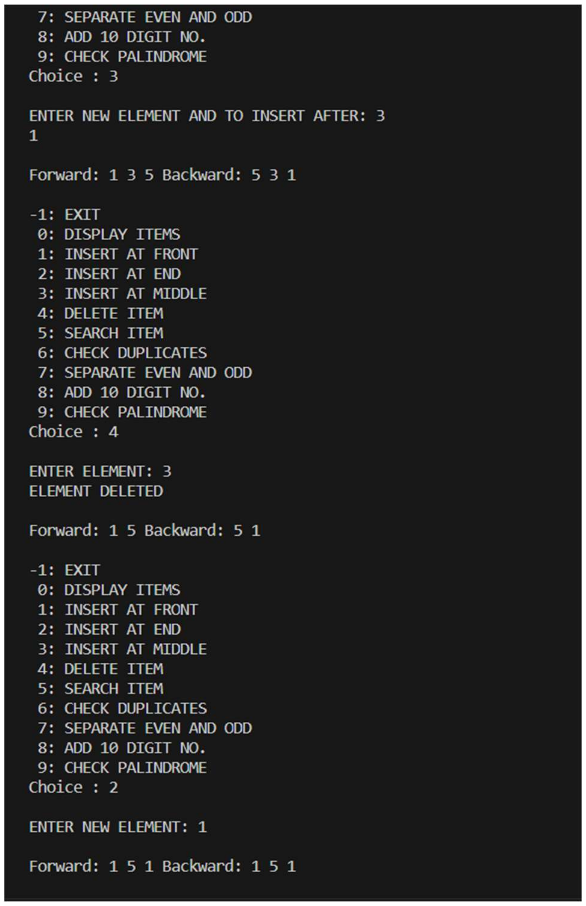
```
                        n2->right=NULL;
                        struct node *sum=(struct node
*)malloc(sizeof(struct node));
                        sum->left=NULL;
                        sum->right=NULL;
                        char num1[11],num2[11];
                        printf("Number 1 : ");
                        scanf("%10s", num1);
                        printf("Number 2 : ");
                        scanf("%10s", num2);
                        for(int i=0;i<10;i++)
                        {
                              insertEnd(n1,(num1[i]-'0'));
                        }
                        for(int i=0;i<10;i++)
                        {
                              insertEnd(n2,(num2[i]-'0'));
                        }
                        printf("Number 1 : ");
                        displayItems(n1);
                              printf("\n");
                        printf("Number 2 : ");
                        displayItems(n2);
                              printf("\n");
                        sum=add10Digit(n1,n2);
                        printf("Sum : ");
                        displayItems(sum);
                              printf("\n");
                        break;
                  }
                  case 9:
                  {
                        if(palindrome(header))
```

```
                                        printf("\nPalindrome.");
                                   else
                                        printf("\nNot Palindrome.");
                                   printf("\n");
                              break;
                         }
                         default:
                         {
                              printf("\nINVALID CHOICE");
                         }
                    }
               }
          }
```

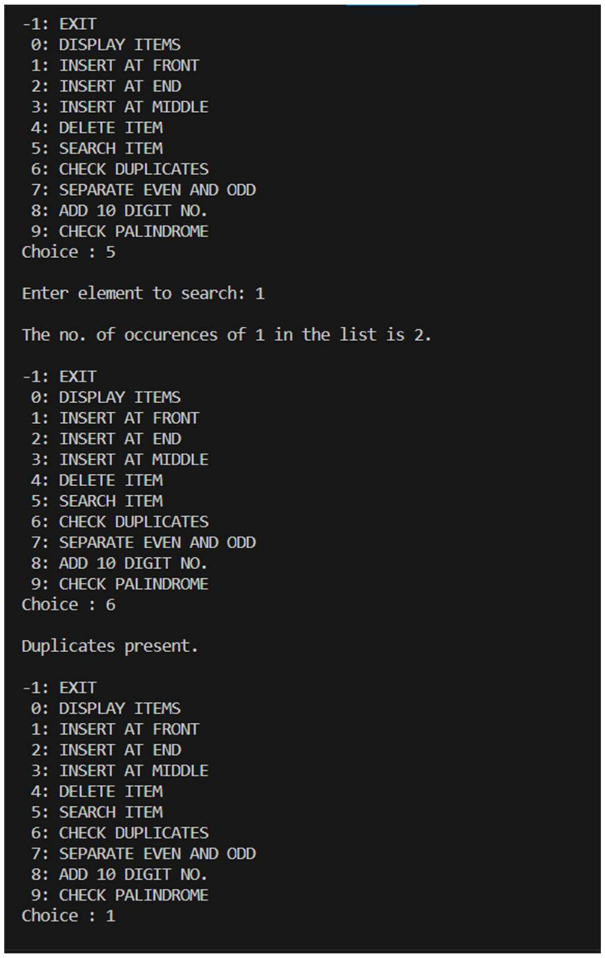
Output Screen:



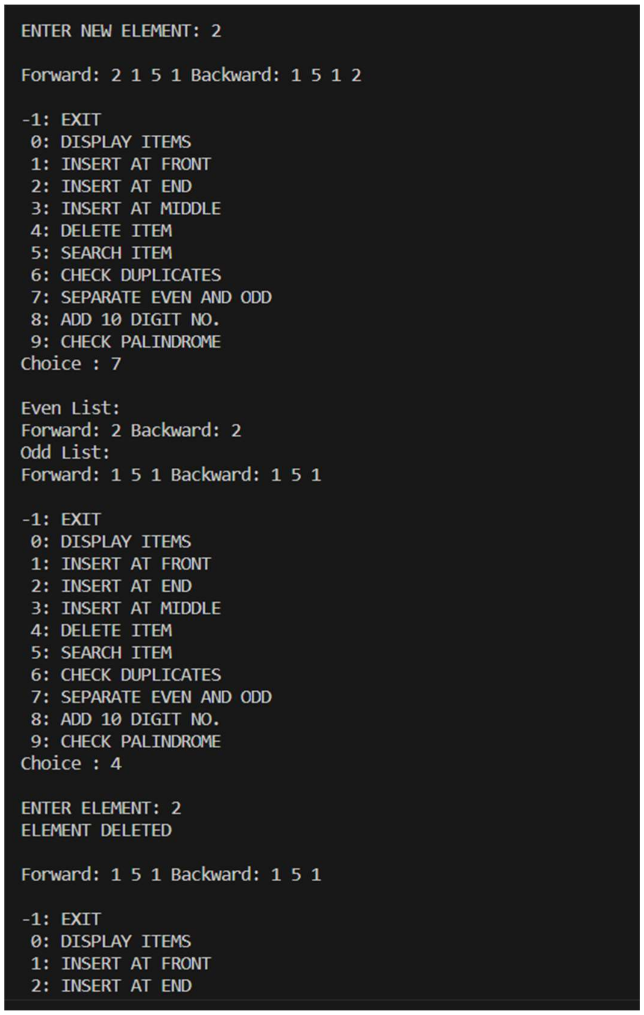
```
PS D:\College\Sem 3\Data Structures\Assignment 3> gcc main.c
PS D:\College\Sem 3\Data Structures\Assignment 3> ./a.exe

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 1

ENTER NEW ELEMENT: 1

Forward: 1 Backward: 1

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 2

ENTER NEW ELEMENT: 5

Forward: 1 5 Backward: 5 1

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
```



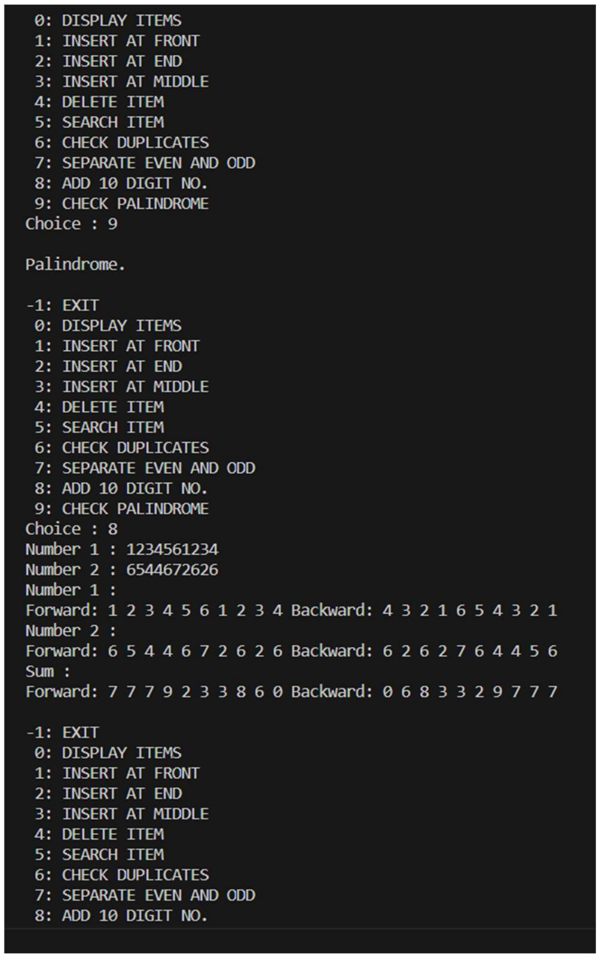
```
  7: SEPARATE EVEN AND ODD
  8: ADD 10 DIGIT NO.
  9: CHECK PALINDROME
Choice : 3

ENTER NEW ELEMENT AND TO INSERT AFTER: 3
1

Forward: 1 3 5 Backward: 5 3 1

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 4

ENTER ELEMENT: 3
ELEMENT DELETED

Forward: 1 5 Backward: 5 1

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 2

ENTER NEW ELEMENT: 1

Forward: 1 5 1 Backward: 1 5 1
```



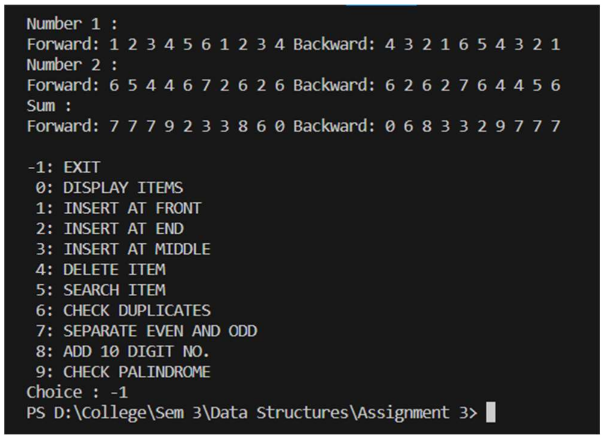
```
-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 5

Enter element to search: 1

The no. of occurences of 1 in the list is 2.

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 6

Duplicates present.

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 1
```



```
ENTER NEW ELEMENT: 2

Forward: 2 1 5 1 Backward: 1 5 1 2

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 7

Even List:
Forward: 2 Backward: 2
Odd List:
Forward: 1 5 1 Backward: 1 5 1

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 4

ENTER ELEMENT: 2
ELEMENT DELETED

Forward: 1 5 1 Backward: 1 5 1

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
```



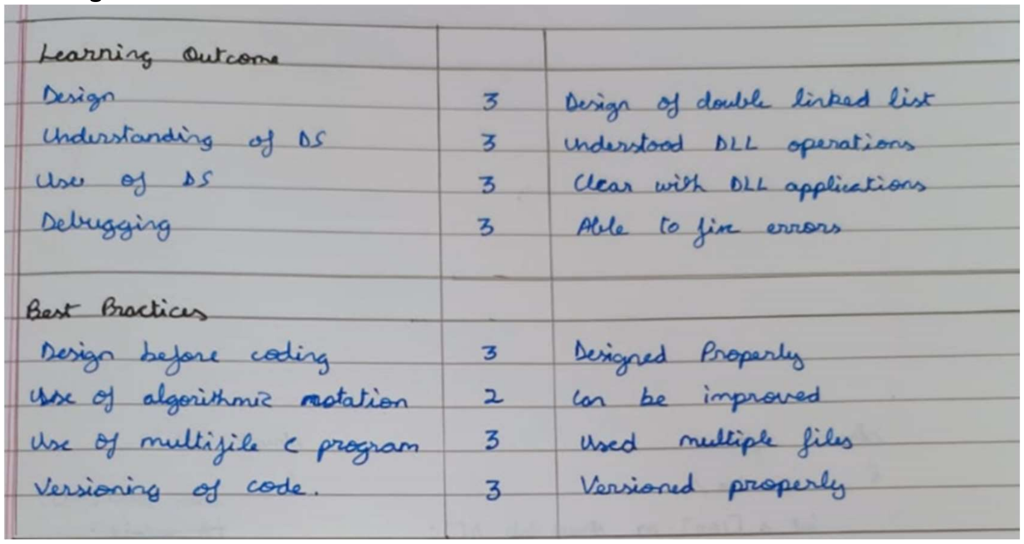
```
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 9

Palindrome.

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : 8
Number 1 : 1234561234
Number 2 : 6544672626
Number 1 :
Forward: 1 2 3 4 5 6 1 2 3 4 Backward: 4 3 2 1 6 5 4 3 2 1
Number 2 :
Forward: 6 5 4 4 6 7 2 6 2 6 Backward: 6 2 6 2 7 6 4 4 5 6
Sum :
Forward: 7 7 7 9 2 3 3 8 6 0 Backward: 0 6 8 3 3 2 9 7 7 7

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
```



```
Number 1 :
Forward: 1 2 3 4 5 6 1 2 3 4 Backward: 4 3 2 1 6 5 4 3 2 1
Number 2 :
Forward: 6 5 4 4 6 7 2 6 2 6 Backward: 6 2 6 2 7 6 4 4 5 6
Sum :
Forward: 7 7 7 9 2 3 3 8 6 0 Backward: 0 6 8 3 3 2 9 7 7 7

-1: EXIT
 0: DISPLAY ITEMS
 1: INSERT AT FRONT
 2: INSERT AT END
 3: INSERT AT MIDDLE
 4: DELETE ITEM
 5: SEARCH ITEM
 6: CHECK DUPLICATES
 7: SEPARATE EVEN AND ODD
 8: ADD 10 DIGIT NO.
 9: CHECK PALINDROME
Choice : -1
PS D:\College\Sem 3\Data Structures\Assignment 3>
```

Date: 19.09.2023 Name: Niranjan.B
Ex. No: 3 Reg. No.: 3122225001082

Learning Outcome:



Learning Outcome		
Design	3	Design of double linked list
Understanding of DS	3	Understood DLL operations
Use of DS	3	Clear with DLL applications
Debugging	3	Able to fix errors
Best Practices		
Design before coding	3	Designed Properly
Use of algorithmic notation	2	Can be improved
Use of multifile c program	3	Used multiple files
Versioning of code.	3	Versioned properly