University of Engineering and Technology, Lahore

Department of Electrical Engineering

Submitted by: Manish Kumar Mahato

Registration number: 2018ee192

Submitted to: Sir Khalid Butt

Section B

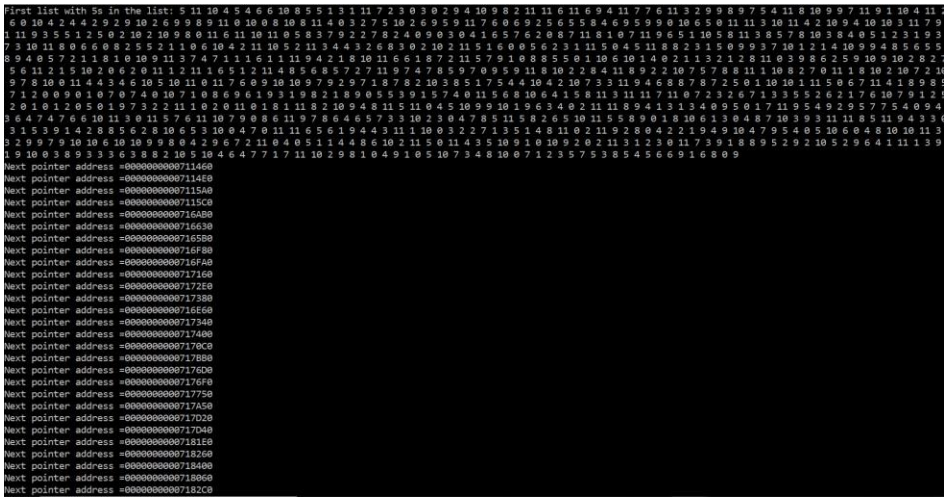
Code Implementation:

```
int main( )
{
    List L,L1;
    Position P,P1,P2,P3;
    int num,i;
    int count=0;    //initialize the counter
    L = MakeEmpty( NULL );  //create an empty list
    P = Header( L );
    for(num=1;num<=1000;num++){
        Insert(rand()%12,L,P);  //insert 1000 random elements into the link list
        P=Advance(P);   //insert each element in next position
    }
    printf("First list with 5s in the list: ");
    PrintList(L);   //print the list
    while( Find(5,L)!=NULL )    //keep on searching for 5 until no 5 is left in the list
        {
        P=Find(5,L);    //get position of 5 in the list
        printf("Next pointer address =%p\n",Advance(P));    //print the next element address
        Delete(5,L);    //delete 5 from the list
        count++;    //increment the counter on each deletion
        }
    printf("First List with deleted 5s\n");
    PrintList(L);  //print list after deleting all the 5s
    printf("Number of deleted entries of 5=%d\n",count);   //print number of deleted 5s
    L1 = MakeEmpty( NULL ); //create another empty list
```

```
    printf("Number of deleted entries of 5=%d\n",count);   //print number of deleted 5s
    L1 = MakeEmpty( NULL ); //create another empty list
    P1 = Header( L1 );
   for(i=1;i<=1000;i++){
        Insert(rand()%12,L1,P1);    //insert 1000 random elements into the link list
        P1=Advance(P1);     //insert each element in next position

    }
    printf("Second list without 4s and 6s ");
    PrintList(L1);//print the list
    P1=Header(L1);

    do

    {   P1=Advance(P1);
        if(Retrieve(P1)==5){    //if 5 is found in the list
        P3=RetrievePrevious(P1);
        Insert(4,L1,P3);    //insert 4 previous to it
        Insert(6,L1,P1);    //insert 6 after it

        }
    } while(!IsLast(P1,L1) );   //search the complete list
    printf("Second List with 5 in between 4 and 6 \n");
    PrintList(L1);  //print the list after inserting 6s and 4s
    return 0;
}
```

```
void
Delete( ElementType X, List L )
{
    Position P, TmpCell;

    P = FindPrevious( X, L );   //find the previous position

    if( !IsLast( P, L ) )  /* Assumption of header use */
    {                      /* X is found; delete it */
        TmpCell = P->Next;
        P->Next = TmpCell->Next;  /* Bypass deleted cell */
        free( TmpCell );    //free the space
        P->Next->Previous=P;//update previous pointer of the next element of the element being deleted
    }
}
*/


    Position
    FindPrevious( ElementType X, List L )
    {
        Position P;

*/      P = L;
*/      while( P->Next != NULL && P->Element != X ) //find element position
*/          P = P->Next;    //keep on searching till end of list or the element is found

*/      return P->Previous; //return the previous pointer on that position
    }
UD */


    void
    Insert( ElementType X, List L, Position P )
    {
        Position TmpCell;

        TmpCell = malloc( sizeof( struct Node ) );  //reserve space for node
        if( TmpCell == NULL )
            FatalError( "Out of space!!!" );
        TmpCell->Element = X;   //store element
        TmpCell->Next = P->Next;    //store next pointer address
        P->Next = TmpCell;  //update previous element next pointer address
        TmpCell->Previous=P;    //store the previous element pointer
        if(TmpCell->Next!=NULL) //check if it is last element
            TmpCell->Next->Previous=TmpCell;    //update next elements previous pointer


}
Position
RetrievePrevious(Position P){
    return P->Previous;     //returns pointer to previous element on that position
}
```

Outputs:



```
Next pointer address =000000000071E560
First List with deleted 5s
11 10 4 4 6 6 10 8 1 3 1 11 7 2 3 0 3 0 2 9 4 10 9 8 2 11 11 6 11 6 9 4 11 7 7 6 11 3 2 9 9 8 9 7 4 11 8 10 9 9 7 11 9 1 10 4 11 2 6 0 10 4 2 4 4 2 9 2 9 10 2 6 9 9 8 9 11
0 10 0 8 10 8 11 4 0 3 2 7 10 2 6 9 9 11 7 6 0 6 9 2 6 8 4 6 9 9 9 0 10 6 0 11 11 3 10 11 4 2 10 9 4 10 10 3 11 7 9 1 11 9 3 1 2 0 2 10 2 10 9 8 0 11 6 11 10 11 0 8 3 7 9 2
 2 7 8 2 4 0 9 0 3 0 4 1 6 7 6 2 0 8 7 11 8 1 0 7 11 9 6 1 10 8 11 3 8 7 8 10 3 8 4 0 1 2 3 1 9 3 7 3 10 11 8 0 6 6 0 8 2 2 1 1 0 6 10 4 2 11 10 2 11 3 4 4 3 2 6 8 3 0 2 10
 2 11 1 6 0 0 6 2 3 1 11 0 4 11 8 8 2 3 1 0 9 9 3 7 10 1 2 1 4 10 9 9 4 8 6 8 9 4 0 7 2 1 1 8 1 0 10 9 11 3 7 4 7 1 1 1 6 1 1 11 9 4 2 1 8 10 11 6 6 1 8 7 2 11 7 9 1 0 8 8
0 1 10 6 10 1 4 0 2 1 1 3 2 1 2 8 11 0 3 9 8 6 2 9 10 9 10 2 8 2 7 6 11 2 1 10 2 0 6 2 0 11 1 2 11 1 6 1 2 11 4 8 6 8 7 2 7 11 9 7 4 7 8 9 7 0 9 9 11 8 10 2 2 8 4 11 8 9 2
2 10 7 7 8 8 11 1 10 8 2 7 0 11 1 8 10 2 10 7 2 10 9 7 8 10 0 11 4 4 3 4 6 10 10 11 0 11 7 6 0 9 10 10 9 7 9 2 9 7 1 8 7 8 2 10 3 8 1 7 4 4 10 4 2 10 7 3 3 11 9 4 6 8 8 7 8
 7 2 0 1 10 10 1 11 0 6 7 11 4 1 8 9 8 7 1 2 0 0 9 0 1 0 7 0 7 4 0 10 7 1 0 8 6 9 6 1 9 3 1 9 8 2 1 8 9 0 3 9 1 7 4 0 11 6 8 10 6 4 1 8 11 3 11 11 7 11 0 7 2 3 2 6 7 1 3 3
2 6 2 1 7 6 10 7 9 1 2 2 0 1 0 1 2 0 0 1 9 7 3 2 2 11 1 0 2 0 11 0 1 8 1 11 8 2 10 9 4 8 11 11 0 4 10 9 9 10 1 9 6 3 4 0 2 11 11 8 9 4 1 3 1 3 4 0 9 0 1 7 11 9 4 9 2 9 7 7
4 0 9 4 3 6 4 7 4 7 6 6 10 11 3 0 11 7 6 11 10 7 9 0 8 6 11 9 7 8 6 4 6 7 3 3 10 2 3 0 4 7 8 11 8 2 6 10 11 8 9 0 1 8 10 6 1 3 0 4 8 7 10 3 9 3 11 11 8 11 9 4 3 3 0 3 1 3 9
 1 4 2 8 8 6 2 8 10 6 3 10 0 4 7 0 11 11 6 6 1 9 4 4 3 11 1 10 0 3 2 2 7 1 3 1 4 8 11 0 2 11 9 2 8 0 4 2 2 1 9 4 9 10 4 7 9 4 0 10 6 0 4 8 10 10 11 3 3 2 9 9 7 9 10 10 6 10
 10 9 9 8 0 4 2 9 6 7 2 11 0 4 0 1 1 4 4 8 6 10 2 11 0 11 4 3 10 9 1 0 10 9 2 0 2 11 3 1 2 3 0 11 7 3 9 1 8 8 9 2 9 2 10 2 9 6 4 1 11 1 3 9 1 9 10 0 3 8 9 3 3 3 6 3 8 8 2 1
0 10 4 6 4 7 7 1 7 11 10 2 9 8 1 0 4 9 1 0 10 7 3 4 8 10 0 7 1 2 3 7 3 8 4 6 6 9 1 6 8 0 9
Number of deleted entries of 5=92
Second list without 4s and 6s 3 4 3 4 9 5 6 0 4 6 10 11 9 8 8 10 9 10 9 4 5 3 10 5 0 1 9 10 7 3 6 4 6 10 11 7 9 4 4 2 6 1 4 0 2 5 8 4 10 1 8 8 3 8 7 6 11 2 2 4 11 8 9 3 10
11 6 6 7 2 10 4 10 5 2 10 6 8 5 1 3 10 2 3 6 2 0 8 6 3 4 0 10 0 7 3 2 3 4 1 2 1 4 3 8 2 9 5 10 1 4 8 4 3 10 11 11 0 5 2 6 10 2 6 8 9 10 3 3 11 1 7 3 8 10 6 5 8 9 6 1 8 9 9
6 3 6 5 5 2 9 4 1 6 9 4 6 0 7 5 7 9 5 8 5 7 8 10 0 4 9 5 8 1 8 10 0 6 4 0 9 0 2 3 7 7 9 9 7 9 1 1 4 10 7 5 0 10 9 6 9 0 3 9 1 9 8 4 10 3 0 10 10 6 6 7 10 8 6 11 3 4 10 8 10
 7 8 6 0 1 10 11 1 9 9 3 6 6 11 10 9 4 8 3 0 8 6 3 0 11 0 3 11 10 8 2 11 10 5 10 6 4 3 4 7 8 4 2 3 5 7 2 3 0 11 8 3 6 10 4 4 7 10 0 2 9 4 6 9 9 3 10 5 0 0 7 0 7 4 11 2 7 5
8 7 11 8 1 4 9 10 9 7 1 1 6 0 11 10 9 8 2 5 9 9 11 1 5 10 8 8 6 1 2 5 5 6 6 0 1 6 3 5 7 7 11 9 11 6 2 6 6 6 7 2 8 11 2 1 8 7 10 6 6 8 0 2 9 6 11 11 1 6 9 5 10 6 10 6 9 11 8
 11 10 1 7 5 11 2 6 9 8 10 8 0 8 1 8 10 7 10 7 5 4 2 8 0 5 8 2 8 0 7 8 7 8 0 5 11 1 7 9 9 8 0 0 7 3 6 1 11 8 6 6 5 11 8 6 7 0 4 11 11 1 11 6 4 1 4 3 0 6 4 9 1 1 4 1 3 0 10
1 2 3 11 1 1 9 0 5 2 4 2 6 6 11 6 5 3 5 0 0 0 8 7 8 9 11 10 3 0 2 6 11 11 9 4 2 11 3 0 0 0 5 10 9 6 4 4 11 8 7 0 0 6 8 11 4 4 1 2 6 1 6 1 7 6 9 5 7 0 4 6 2 7 2 1 6 3 5 9 7
2 4 9 7 0 1 9 0 4 11 1 2 4 3 7 9 6 4 11 6 8 6 5 9 1 0 6 11 1 4 10 9 7 11 9 1 0 0 8 7 7 2 10 11 11 6 0 5 5 5 5 9 2 6 6 4 3 3 9 2 11 1 6 5 9 5 2 2 7 0 4 11 10 10 10 1 7 3 9 1
1 9 10 7 7 4 10 4 7 9 2 6 2 11 0 10 0 9 5 4 11 1 3 6 11 1 2 4 5 4 1 5 11 11 4 6 3 0 10 3 2 2 1 10 3 11 4 5 8 7 0 6 10 6 5 1 3 11 11 4 1 1 1 5 2 2 5 5 3 11 2 7 5 8 9 9 0 0 1
 3 10 3 9 8 8 5 8 5 11 10 1 10 10 4 10 3 10 0 5 0 6 1 4 1 8 5 10 5 7 6 0 0 2 8 1 2 6 7 1 2 11 0 10 7 9 7 1 6 4 11 7 11 4 2 11 4 7 6 7 0 7 8 1 1 1 3 0 5 10 4 6 9 8 2 5 2 4 1
1 5 3 1 1 0 7 9 9 9 9 10 11 5 1 0 0 3 1 5 4 1 0 6 8 7 9 8 0 0 3 9 4 11 0 7 4 0 7 9 7 11 7 1 0 7 5 6 4 3 0 0 11 1 3 4 11 0 10 0 9 0 7 10 1 2 8 0 9 2 8 0 0 4 10 5 0 2 2 5 5 0
 7 8 0 11 11 2 1 8 1 8 7 4 1 1 4 4 7 6 11 10 9 6 8 8 2 8 5 6 10 6 11 3 1 7 11 8 7 7 5 11 1 5 4 5 9 1 3 10 9 7 8 10 6 4 7 5 0 0 5 2 5 11 10 7 2 0 11 0 11 2 8 2 3 5 1 6 11 9
3 10 11 2 2 11 3 4 5 0 11 3 9 6 2 5 9 1 11 3 5 2 7 5 2 5 1 10 6 0 4 7 4 8 1 11 2 8 5 11 1 4 7 11 9 7 5 1 9 9 10 3 1 10 0 3 11 3 9 2 3 10
Second List with 5 in between 4 and 6
3 4 3 4 9 4 5 6 6 0 4 6 10 11 9 8 8 10 9 10 9 4 4 5 6 3 10 4 5 6 0 1 9 10 7 3 6 4 6 10 11 7 9 4 4 2 6 1 4 0 2 4 5 6 8 4 10 1 8 8 3 8 7 6 11 2 2 4 11 8 9 3 10 11 6 6 7 2 10
4 10 4 5 6 2 10 6 8 4 5 6 1 3 10 2 3 6 2 0 8 6 3 4 0 10 0 7 3 2 3 4 1 2 1 4 3 8 2 9 4 5 6 10 1 4 8 4 3 10 11 11 0 4 5 6 2 6 10 2 6 8 9 10 3 3 11 1 7 3 8 10 6 4 5 6 8 9 6 1
8 9 9 6 3 6 4 5 6 4 5 6 2 9 4 1 6 9 4 6 0 7 4 5 6 7 9 4 5 6 8 4 5 6 7 8 10 0 4 9 4 5 6 8 1 8 10 0 6 4 0 9 0 2 3 7 7 9 9 7 9 1 1 4 10 7 4 5 6 0 10 9 6 9 0 3 9 1 9 8 4 10 3 0
 10 10 6 6 7 10 8 6 11 3 4 10 8 10 7 8 6 0 1 10 11 1 9 9 3 6 6 11 10 9 4 8 3 0 8 6 3 0 11 0 3 11 10 8 2 11 10 4 5 6 10 6 4 3 4 7 8 4 2 3 4 5 6 7 2 3 0 11 8 3 6 10 4 4 7 10
0 2 9 4 6 9 9 3 10 4 5 6 0 0 7 0 7 4 11 2 7 4 5 6 8 7 11 8 1 4 9 10 9 7 1 1 6 0 11 10 9 8 2 4 5 6 9 9 11 1 4 5 6 10 8 8 6 1 2 4 5 6 4 5 6 6 6 0 1 6 3 4 5 6 7 7 11 9 11 6 2
6 6 6 7 2 8 11 2 1 8 7 10 6 6 8 0 2 9 6 11 11 1 6 9 4 5 6 10 6 10 6 9 11 8 11 10 1 7 4 5 6 11 2 6 9 8 10 8 0 8 1 8 10 7 10 7 4 5 6 4 2 8 0 4 5 6 8 2 8 0 7 8 7 8 0 4 5 6 11
1 7 9 9 8 0 0 7 3 6 1 11 8 6 6 4 5 6 11 8 6 7 0 4 11 11 1 11 6 4 1 4 3 0 6 4 9 1 1 4 1 3 0 10 1 2 3 11 1 1 9 0 4 5 6 2 4 2 6 6 11 6 4 5 6 3 4 5 6 0 0 0 8 7 8 9 11 10 3 0 2
6 11 11 9 4 2 11 3 0 0 0 4 5 6 10 9 6 4 4 11 8 7 0 0 6 8 11 4 4 1 2 6 1 6 1 7 6 9 4 5 6 7 0 4 6 2 7 2 1 6 3 4 5 6 9 7 2 4 9 7 0 1 9 0 4 11 1 2 4 3 7 9 6 4 11 6 8 6 4 5 6 9
1 0 6 11 1 4 10 9 7 11 9 1 0 0 8 7 7 2 10 11 11 6 0 4 5 6 4 5 6 4 5 6 4 5 6 9 2 6 6 4 3 3 9 2 11 1 6 4 5 6 9 4 5 6 2 2 7 0 4 11 10 10 10 1 7 3 9 11 9 10 7 7 4 10 4 7 9 2 6
2 11 0 10 0 9 4 5 6 4 11 1 3 6 11 1 2 4 4 5 6 4 1 4 5 6 11 11 4 6 3 0 10 3 2 2 1 10 3 11 4 4 5 6 8 7 0 6 10 6 4 5 6 1 3 11 11 4 1 1 1 4 5 6 2 2 4 5 6 4 5 6 3 11 2 7 4 5 6 8
 9 9 0 0 1 3 10 3 9 8 8 4 5 6 8 4 5 6 11 10 1 10 10 4 10 3 10 0 4 5 6 0 6 1 4 1 8 4 5 6 10 4 5 6 7 6 0 0 2 8 1 2 6 7 1 2 11 0 10 7 9 7 1 6 4 11 7 11 4 2 11 4 7 6 7 0 7 8 1
1 1 3 0 4 5 6 10 4 6 9 8 2 4 5 6 2 4 11 4 5 6 3 1 1 0 7 9 9 9 9 10 11 4 5 6 1 0 0 3 1 4 5 6 4 1 0 6 8 7 9 8 0 0 3 9 4 11 0 7 4 0 7 9 7 11 7 1 0 7 4 5 6 6 4 3 0 0 11 1 3 4 1
1 0 10 0 9 0 7 10 1 2 8 0 9 2 8 0 0 4 10 4 5 6 0 2 2 4 5 6 4 5 6 0 7 8 0 11 11 2 1 8 1 8 7 4 1 1 4 4 7 6 11 10 9 6 8 8 2 8 4 5 6 6 10 6 11 3 1 7 11 8 7 7 4 5 6 11 1 4 5 6 4
 4 5 6 9 1 3 10 9 7 8 10 6 4 7 4 5 6 0 0 4 5 6 2 4 5 6 11 10 7 2 0 11 0 11 2 8 2 3 4 5 6 1 6 11 9 3 10 11 2 2 11 3 4 4 5 6 0 11 3 9 6 2 4 5 6 9 1 11 3 4 5 6 2 7 4 5 6 2 4 5
```