SUNWAY
INT'L BUSINESS SCHOOL

Programme Name: __________**BCS HONS**____

Course Code: ___**CSC 2516**_________

Course Name: _________**Data Structure and Algorithm**____________________________________

Internal Examination

Date of Submission: ________**8/24/2021**______________

Submitted By:

Student Name**: Dipesh Tha Shrestha**

IUKL ID: **041902900028**

Semester**: Fourth Semester**

Intake**: September 2019**

Submitted To:

Faculty Name**: Prakash Chandra Sir**

Department**: LMS**

1. **Explain data structures and its importance. Also list at least 5 data structures.**

Answer: A data structure is a collection of data elements that provides an efficient method for storing and organizing data in a computer so that it can be used effectively.

Data Structures are important because they allow programmers to work with data efficiently. Data structure provides the right way to organize information in the digital space.

List of 5 data structures are given below:

- Array
- List
- Float
- Integer
- Pointers

2. **Explain the working of linear search algorithm and analyze its efficiency**

Answer: A linear search algorithm is used to locate a target value within a list. It checks each element of the list for the target value in a sequential manner until a match is found or all elements have been searched.

In linear search, we search for an element or value in an array by traversing it from the beginning until the element or value we want is found. It compares the element to be searched with all of the elements in the array and returns the index of the element in the array if the element is successfully matched, otherwise it returns -1.

Best Case Scenario

It's possible that the element you're looking for is in the first position.

In this case, the search is successful after only one comparison.

As a result, the linear search algorithm takes O(1) operations in the best case scenario.

Worst Case Scenario

The element being searched may be at the very end of the array, or it may not exist at all.

In the first case, the search is successful after n comparisons.
With n comparisons, the search ends in failure in the latter case.
As a result, in the worst-case scenario, the linear search algorithm requires O(n) operations.

Linear Search efficiency
The efficiency of a method is determined by the amount of time it takes to search a record in a search table or the number of comparisons it makes. Only one comparison is made if the desired record is found in the first position of the search table. When the desired record is the most recent, n comparisons must be performed.

The average number of comparisons will be (n+1/2) if the record appears somewhere in the search table. The worst-case efficiency of this technique is O(n), where n is the execution order.

3. For doubly linked list:
   a. Write C program for Insertion, Deletion and Search operation uses in real world applications.

```
#include <stdio.h>
#include <stdlib.h>
//structure of Node with prev and next pointers
struct node {
int data;
struct node * prev;
struct node * next;
}*head, *last;

void createList(int n);
void displayList();
void insert_given_pos(int data, int position);

int main()
{
int n, data, pos;
head = NULL;
last = NULL;
```

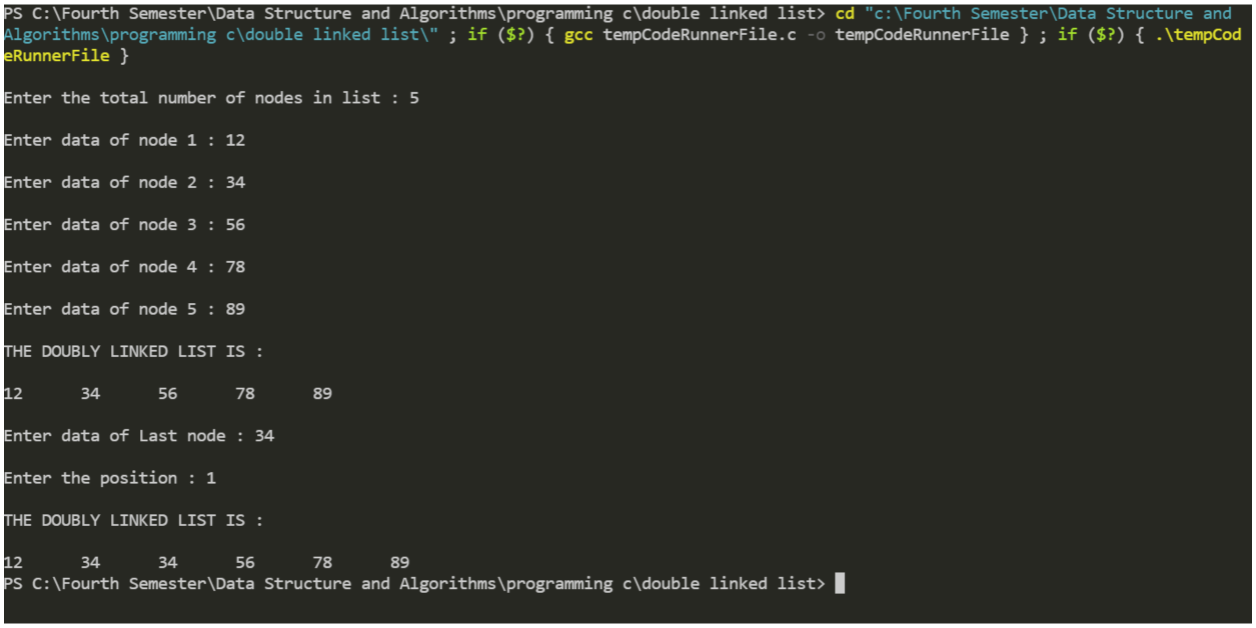
```c
printf("\nEnter the total number of nodes in list : "); // Input the number of no
des
scanf("%d", &n);
createList(n);
displayList();
printf("\n\nEnter data of Last node : ");
scanf("%d", &data);
printf("\nEnter the position : ");
scanf("%d",&pos);
insert_given_pos(data, pos);
displayList();
return 0;
}

void createList(int n)
{
int i, data;
struct node *newNode;
if(n >= 1)
{
head = (struct node *)malloc(sizeof(struct node));
printf("\nEnter data of node 1 : ");
scanf("%d", &data);
head->data = data;
head->prev = NULL; // HEAD nodes's prev is set to NULL
head->next = NULL; // HEAD nodes's next is set to NULL
last = head;
for(i=2; i<=n; i++)
{
newNode = (struct node *)malloc(sizeof(struct node));
printf("\nEnter data of node %d : ", i);
scanf("%d", &data);
newNode->data = data;
newNode->prev = last; // Link new node with the previous node
newNode->next = NULL;
last->next = newNode; // Link previous node with the new node
last = newNode; // Make new node as last node
}}}

void displayList()
{
struct node * temp;
int n = 1;
if(head == NULL)
{
```

```
printf("\nList is empty.\n");
}
else
{
temp = head;
printf("\nTHE DOUBLY LINKED LIST IS :\n\n");
while(temp != NULL)
{
printf("%d\t", temp->data);
n++;
/* Move the current pointer to next node */
temp = temp->next;
}
}
}

void insert_given_pos(int data, int position)
{
int i;
struct node * newNode, *temp;
if(head == NULL)
{
printf("Error, List is empty!\n");
}
else
{
temp = head;
i=0;
while(i<position-1 && temp!=NULL)
{
temp = temp->next;
i++;
}
if(temp!=NULL)
{
newNode = (struct node *)malloc(sizeof(struct node));
newNode->data = data;
newNode->next = temp->next; // Connect new node with n + 1th node
newNode->prev = temp; // Connect new node with n - 1th node
if(temp->next != NULL)
{
/* Connect pos+1th node with new node */
temp->next->prev = newNode;
}
/* Connect pos-1th node with new node */
temp->next = newNode;
}
else
{
printf("Error, Invalid position\n");
}}}
```



```
PS C:\Fourth Semester\Data Structure and Algorithms\programming c\double linked list> cd "c:\Fourth Semester\Data Structure and 
Algorithms\programming c\double linked list\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if ($?) { .\tempCod
eRunnerFile }

Enter the total number of nodes in list : 5

Enter data of node 1 : 12

Enter data of node 2 : 34

Enter data of node 3 : 56

Enter data of node 4 : 78

Enter data of node 5 : 89

THE DOUBLY LINKED LIST IS :

12      34      56      78      89

Enter data of Last node : 34

Enter the position : 1

THE DOUBLY LINKED LIST IS :

12      34      34      56      78      89
PS C:\Fourth Semester\Data Structure and Algorithms\programming c\double linked list> 
```

4. For the given list of numbers:
[15, 8 , 25 , 40 , 7 , 9 , 10]

Create binary search tree.

pesh Jha Shrestha CSC_2516

④ For the given list of number
[15, 8; 25, 40, 7, 9, 10]
ⓐ Create binary Search tree.

Solution:

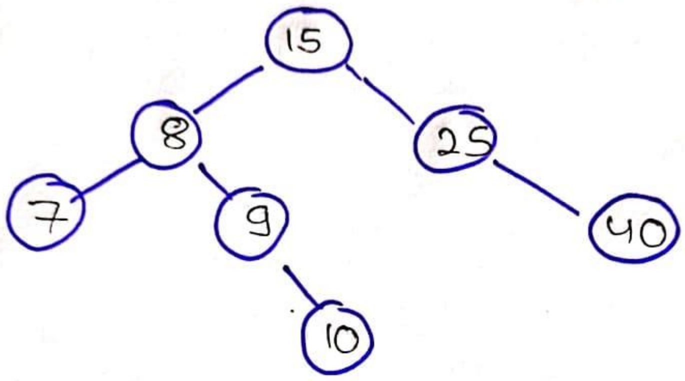


Fig: Binary Search Tree.

ⓑ

Inorder traversal = [7, 8, 9, 10, 15, 25, 40]

Preorder traversal = [15, 8, 7, 9, 10, 15, 25, 40]

Postorder traversal = [7, 10, 9, 8, 40, 25, 15]