Programme Name: **BCS HONS**

Course Code: **CSC 2516**

Course Name: **Data Structure and Algorithm**

Assignment / Lab Sheet / Project / Case Study No. **2**

Date of Submission: **8/10/2021**

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.

a) Explain the concept of Linked List and advantages of Linked list over arrays.

Answer: A linked list is a linear data structure that includes a series of connected nodes. Here, each node stores the data and the address of the next node. Advantage of a linked list over array is that, we can add any number of elements in the list, this is not possible in case of an array. Linked list provide an efficient way of storing related data and perform basic operations such as insertion, deletion and updating of information at the cost of extra space required for storing the address.

b) Write C program to implement: Insert, delete, search and display operations in singly linked list

Answer: C program to implement: Insert, delete, search and display operations in singly linked list is given below:

```
// program to implement insert, delete, search and display singly linked list
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *head=NULL;
void insert_at_begining()
{
    struct node *ptr;
    int item;
    ptr = (struct node*)malloc(sizeof(struct node));
    if (ptr==NULL)
    {
        printf("\noverflow\n");
    }
    else
    {
        printf("\nEnter element to insert: ");
        scanf("%d", &item);
        ptr->data=item;
        ptr->next=NULL;
        if(head!=NULL)
        {
            ptr->next=head;
            head=ptr;
        }
```

```
        else{

        head=ptr;
        }
    }
}
void insert_at_pos()
{
    struct node *ptr;
    struct node *temp;
    int item, i=1, pos;
    ptr = (struct node*)malloc(sizeof(struct node));
    if (ptr==NULL)
    {
        printf("\noverflow\n");
    }
    else
    {
        printf("\nEnter element to be inserted: ");
        scanf("%d", &item);
        ptr->data=item;
        ptr->next=NULL;
        printf("\nEnter position to be inserted:");
        scanf("%d",&pos);
        while (i<pos-1)
        {
            temp= temp->next;
            i++;
        }
        ptr->next = temp->next;
        temp->next = ptr;
    }
}
void insert_at_end()
{
    struct node *ptr;
    struct node *temp;
    int item;
    ptr = (struct node*)malloc(sizeof(struct node));
    if (ptr==NULL)
    {
        printf("\noverflow\n");
    }
    else
    {
```

```
        printf("\nEnter element to insert at last: ");
        scanf("%d", &item);
        ptr->data=item;
        ptr->next=NULL;
        temp=head;
        while(temp->next!=NULL)
        {
            temp = temp->next;
        }
        temp->next = ptr;
        ptr->next = NULL;
    }
}
void delete_from_begining()
{
   struct node *temp;
   struct node *ptr;
   if (head == NULL)
   {
      printf("\nUnderflow\n");
   }
   else
   {
      ptr = head;
      temp = ptr->next;
      head = temp;
      free(ptr);
      printf("\nFirst node deleted successfully\n");
   }
}
void delete_from_pos()
{
    struct node *ptr;
    struct node *temp;
    int i=1, pos;
    ptr = (struct node*)malloc(sizeof(struct node));
    if (ptr==NULL)
    {
        printf("\noverflow\n");
    }
    else
    {
        printf("\nEnter the position: ");
        scanf("%d", &pos);
        temp = head;
```

```
        while(i<pos-1)
        {
            i++;
            temp = temp->next;
        }
        ptr = temp->next;
        temp->next = ptr->next;
        free(ptr);
        printf("\n Node deleted successfully\n");
    }

}
void delete_from_end()
{
    struct node *temp;
    struct node *ptr;
    if(head==NULL)
    {
        printf("\nUnderflow\n");
    }
    else
    {
        temp = head;
        while(temp->next->next!=NULL)
        {
            temp = temp->next;
        }
        ptr = temp->next->next;
        temp->next = NULL;
        free(ptr);
        printf("\n Node deleted successfully\n");
    }
}
void display()
{
    struct node *temp;
    temp = head;
    printf("\nThe list contains following data:\n");
    while(temp!=NULL)
    {
        printf("%d\t",temp->data);
        temp = temp->next;
    }
}
void search()
```

```
{
    struct node *temp;
    int key;
    int i=1;
    if(head!=NULL)
    {
        printf("\nEnter the key to be searched: ");
        scanf("%d",&key);
        temp = head;
        while(temp!=NULL)
        {
            if(temp->data==key)
            {
                printf("\n%d is found at pos: %d\n",key,i);
                break;
            }
            i++;
            temp = temp->next;
        }
    }
    else
    {
        printf("\nlist is empty\n");
    }

}
int main()
{
    insert_at_begining();
    insert_at_pos();
    insert_at_pos();
    insert_at_end();
    search();
    delete_from_begining();
    delete_from_pos();
    delete_from_end();
    display();



}
```

Output;

```
Enter element to insert: 12

Enter element to be inserted: 34

Enter position to be inserted:2

Enter element to be inserted: 34

Enter position to be inserted:1

Enter element to insert at last: 56

Enter the key to be searched: 12

12 is found at pos: 1

First node deleted successfully

Enter the position: 2

 Node deleted successfully

 Node deleted successfully

The list contains following data:
34
```

2. Using STACK, convert the following infix expression into postfix expression:
a) (A+B)*C-(D-E)*(F+G)
NOTE: use tables with proper heading and descriptions

Date
Page

Q No 2

(a)

$(A+B)^* C-(D-E)^*(F+G)$

S.N	Scanned	Stack	Postfix Expression	Description
1		(		Start
2	A	(	A	
3	+	(+	A	
4	B	(+	AB	
5	)	(	AB+	Pop from stack when right Paranthese are encounter
6	*	(*	AB+	
7	C	(*	AB+C	
8	-	(-	AB+C*	(*) has higher precedence
9	(	(-(	AB+C*	
10	D	(-(	AB+C*D	
11	-	(-(-	AB+C*D	
12	E	(-(-	AB+C*DE	
13	)	(-	AB+C*DE-	Pop from stack()
14	*	C-*	AB+C*DE-	
15	(	(-*(	AB+C*DE-	
16	F	(-*(	AB+C*DE-F	
17	+	(-*(+	AB+C*DE-F	
18	G	(-*(+	AB+C*DE-FG	
19	)	(-*	AB+C*DE-FG+	
20	)	EMPTY	AB+C*DE-FG+*-	END

∴ The Postfix of $(A+B)^*C-(D-E)^*(F+G)$
$= AB+C*DE-FG+*-$

3. a) List real world applications of doubly linked lists

Answer: The real world applications of doubly linked lists are as follow:

- It is used by browsers to implement backward and forward navigation of visited web pages i.e. back and forward button.
- It is also used by various applications to implement Undo and Redo functionality.
- It can also be used to represent deck of cards in games.
- It is also used to represent various states of a game.
- It can be used in navigation systems where both front and back navigation is required.

b) Write a C program to implement queue data structure using linked lists.

Answer: C program to implement queue data structure using linked lists are given below:

```c
// implementation of queue using linked list
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *front=NULL, *rear=NULL;
void enqueue()
{
    struct node *ptr,*temp;
    int data;
    ptr = (struct node*)malloc (sizeof(struct node));
    if (ptr==NULL)
    {
        printf("\noverflow\n");
    }
    else
    {
        printf("\nEnter the data to be inserted into Q:\n");
        scanf("%d", &data);
        ptr->data = data;
        ptr->next = NULL;
        if (front==NULL && rear==NULL)
        {
```

```
            front=ptr;
            rear=ptr;
        }
        else
        {
            temp=rear;
            temp->next = ptr;
            rear = ptr;
        }
    }
}
void dequeue()
{
    struct node *temp;
    if (front==NULL)
    {
        printf("\nQueue is empty:underflow\n");
    }
    else
    {
        if (front!=rear)
        {
            temp = front;
            front = temp->next;
            free(temp);
        }
        else
        {
            temp = front;
            front = NULL;
            rear = NULL;
            free (temp);
        }
    }
}

void display()
{
    struct node *temp;
    if (front==NULL && rear==NULL)
    {
        printf("\nQueue is empty\n");
    }
    else
    {
```

```
        temp = front;
        printf("\nThe queue contains following elements:\n");
        while(temp!=NULL)
        {
            printf("%d\t",temp->data);
            temp = temp->next;
        }
    }
}
void main()
{
    int choice;
    char ch='y';
    while(ch=='y')
    {
        printf("\nCHOOSE 1:Enqueue 2:Dequeue 3:Display\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            default:
                printf("\nInvalid Choice\n");
                break;

        getchar();
        printf("\ndo you want to continue(y/n)");
        scanf("%c",&ch);
    }

}
```

Output:

```
CHOOSE 1:Enqueue 2:Dequeue 3:Display
1

Enter the data to be inserted into Q:
34

do you want to continue(y/n)y

CHOOSE 1:Enqueue 2:Dequeue 3:Display
1

Enter the data to be inserted into Q:
45

do you want to continue(y/n)y

CHOOSE 1:Enqueue 2:Dequeue 3:Display
3

The queue contains following elements:
34      45
do you want to continue(y/n)y

CHOOSE 1:Enqueue 2:Dequeue 3:Display
2

do you want to continue(y/n)y

CHOOSE 1:Enqueue 2:Dequeue 3:Display
3

The queue contains following elements:
45
```

Thank you