Report on

"Data Structures"

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering in the course of ***Data Structures*** ***(*****19CS3PCDST*****)***

Submitted by

DHRUV DUBEY
(1BM19CS048)

Under the Guidance of
Dr. Kayarvizhy N.
Associate Professor
Department of CSE

Department of Computer Science and Engineering
BMS College of Engineering
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2020-2021

B M S COLLEGE OF ENGINEERING

P.O. Box No: 1908 Bull Temple Road Bangalore-560019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ASSESSMENT

Report on **Data Structures (19CS3PCDST),** "Advanced Algorithm assignment" has been successfully completed by **Dhruv Dubey** at B.M.S College of Engineering in partial fulfillment of the requirements for the 3rd Semester, degree in Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during academic year 2020-2021.

Dr. Kayarvizhy N.

Associate Professor
Department of Computer science

Final Marks Awarded

Obtained	Total

CONTENT

PROGRAM 1

Write a program to simulate the working of stack using an array with the following:

a) Push

b) Pop

c) Display

The program should print appropriate messages for stack overflow, stack underflow

CODE:

```
#include<stdio.h
>
                    #include<stdbool.h>

                    #define MAX 5
                    int top = -1;

                    void push(int ele,int *stack) {
                          if(top == MAX -1) {
                                printf("Stack overflow\n");
                          }
                          else {
                                top++;
                                stack[top] = ele;
                                printf("%d is pushed into the stack \n",ele);
                          }
                    }

                    bool empty() {
                          if(top==-1) {
                                return true;
                          }
                          return false;
                    }

                    void pop(int *stack) {
                          if(!empty()) {
                                printf("%d popped out\n",stack[top]);
```

```
            top--;
      }

      else
            printf("Stack Underflow\n");

}

void display(int *stack) {
      int i;
      if(top==-1) {
            printf("Stack is empty.\n");
      }
      else {
            for(i=top;i>=0;i--)
            {
                  printf("%d\n",stack[i]);
            }
      }
}


int main() {
      int element;
      int stack1[MAX];
      int stack2[MAX];
      printf("Select from the choices available: \n");
      printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
      int option;
      printf("Enter your choice : ");
      scanf("%d",&option);
      while(option!=4) {
            switch(option) {
                  case 1: printf("Enter element to be pushed:
");
                              scanf("%d",&element);
```

```
                                    push(element,stack1);
                                    break;
                            case 2: pop(stack1);
                                    break;

                            case 3: display(stack1);
                                    break;
                        }
                        printf("Enter your next choice : ");
                        scanf("%d",&option);
                    }
                    return 0;
                }
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air Data_StrC % ./stacks_imp
Select from the choices available:
1.Push  2.Pop   3.Display       4.Exit
Enter your choice : 1
Enter element to be pushed: 51
51 is pushed into the stack
Enter your next choice : 1
Enter element to be pushed: 4
4 is pushed into the stack
Enter your next choice : 1
Enter element to be pushed: 3
3 is pushed into the stack
Enter your next choice : 1
Enter element to be pushed: 2
2 is pushed into the stack
Enter your next choice : 1
Enter element to be pushed: 1
1 is pushed into the stack
Enter your next choice : 1
Enter element to be pushed: 0
Stack overflow
Enter your next choice : 3
1
2
3
4
51
Enter your next choice : 2
1 popped out
Enter your next choice : 2
2 popped out
Enter your next choice : 2
3 popped out
Enter your next choice : 2
4 popped out
Enter your next choice : 2
51 popped out
Enter your next choice : 2
Stack Underflow
Enter your next choice : 3
Stack is empty.
Enter your next choice : 4
dhruvdubey@Dhruvs-MacBook-Air Data_StrC %
```

PROGRAM 2:

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

CODE:

```
#include<stdio.
h>
                    #define Max 100
                    char stack[Max];
                    int top = -1;


                    void push(char ch)
                    {
                          if(top==Max-1) {
                                printf("Stack Full!\n");
                          }
                          else
                          {
                                top++;
                                stack[top] = ch;
                          }
                    }



                    char pop()
                    {
                          char item;
                          if(top==-1)
                                printf("Empty Stack!\n");
                          else
                          {
                                item = stack[top];
                                top--;
                                return item;
                          }
                    }
```

```
int stackempty()
{
      if(top==-1)
            return 1;
      return 0;
}

char stacktop()
{
      if(top==-1) {
            printf("Empty stack\n");
      }
      else
            return stack[top];
}

int priority(char ch)
{
      switch(ch) {
            case '+':
            case '-': return 1;
            case '*':
            case '/': return 2;
            case '^': return 3;
            default:  return 0;
      }
}


int main()
{
      char infix[100];
      int i,item;
      printf("Enter the infix expression: ");
      scanf("%s",infix);
```

```
        printf("Infix Expression: %s\n",infix);
        printf("Postfix Expression: ");
        i=0;
        while(infix[i] != '\0') {
              switch(infix[i]) {
                    case '(' : push(infix[i]);
                                 break;
                    case ')' : while((item = pop())!='(') {
                                       printf("%c",item);

                                    }
                                    break;

                    case '+':
                    case '-':
                    case '*':
                    case '/':
                    case '^':
                                 while(!stackempty()
&&priority(infix[i])<=priority(stacktop()))
                                 {
                                    item = pop();
                                    printf("%c",item);
                                 }
                                 push(infix[i]);
                                 break;

                    default: printf("%c",infix[i]);
                               break;
              }
              i++;
        }
        while(!stackempty()) {
              printf("%c",pop());
        }
```

```
    return 0;
}
```

OUTPUT:

```
Last login: Fri Oct  9 10:31:46 on ttys000
/Users/dhruvdubey/Desktop/Data_StrC/infixToPostfix ; exit;
dhruvdubey@Dhruvs-MacBook-Air ~ % /Users/dhruvdubey/Desktop/Data_StrC/infix
ToPostfix ; exit;
Enter the infix expression: A+(B*C)^(D+Z)-F
Infix Expression: A+(B*C)^(D+Z)-F
Postfix Expression: ABC*DZ+^+F-
[Process completed]
```

PROGRAM 3:

WAP to simulate the working of a queue of integers using an array. Provide the following operations:

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions.

CODE:

```
#include<stdio.
h>
                #define Max 5
                //intializing front and rear
                int front = 0;
                int rear = -1;
                //intializing queue
                int queue[Max];


                //Func to push into queue
```

```
void push(int d) {
      if(rear == Max - 1)
            printf("Queue is full\n");

      else
     {
       rear++;
       queue[rear]=d;

     }
}

//Func to delete from queue
int pop()
{
     int item;
     if(front == -1)
         return -1;
     else
     {
         item=queue[front];
         front++;
         if(front>rear)
         {
             front=-1;
             rear=-1;
         }
         return item;
     }

}

//Func to display
void display() {
      if(front==-1) {
```

```
            printf("No elements present in the queue to
display!\n");
            return;
      }

      else {
            for(int i=front;i<=rear;i++) {
                  printf("%d ",queue[i]);
            }
            printf("\n");
      }
}

int main() {
      int choice;
      int ele;
      printf("1.Insert in Queue\n 2.Delete from Queue\n
3.Display Queue\n 4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&choice);


      while(choice!=4) {
            switch(choice) {
                  case 1:
                        printf("Enter the element: ");
                        scanf("%d",&ele);
                        push(ele);
                        break;

                  case 2:
                        ele = pop();
                        if(ele==-1) {
                              printf("Queue is empty!\n");
                              break;
                        }
```

```
                        printf("%d is popped out from
queue\n",ele);
                        break;

                case 3:
                        display();
                        break;
            }
            printf("Enter your choice: ");
            scanf("%d",&choice);
        }
        printf("Exit \n");
        return 0;
}
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air Data_StrC % ./linearQueue
1.Insert in Queue
2.Delete from Queue
3.Display Queue
4.Exit
Enter your choice: 1
Enter the element: 1
Enter your choice: 1
Enter the element: 2
Enter your choice: 1
Enter the element: 3
Enter your choice: 1
Enter the element: 4
Enter your choice: 1
Enter the element: 5
Enter your choice: 1
Enter the element: 6
Queue is full
Enter your choice: 3
1 2 3 4 5
Enter your choice: 2
1 is popped out from queue
Enter your choice: 2
2 is popped out from queue
Enter your choice: 2
3 is popped out from queue
Enter your choice: 2
4 is popped out from queue
Enter your choice: 2
5 is popped out from queue
Enter your choice: 2
Queue is empty!
Enter your choice: 3
No elements present in the queue to display!
Enter your choice: 4
Exit
dhruvdubey@Dhruvs-MacBook-Air Data_StrC %
```

PROGRAM 4:

2 WAP to simulate the working of a circular queue of integers using an array. Provide the following operations. a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

CODE:

```
#include<stdio.
h>
                  #define MAX 5
                  int queue[MAX];
                  int front = -1;
                  int rear = -1;

                  void enqueue(int data) {
                        if((front == 0 && rear == MAX-1) || front == rear+1) {
                              printf("Queue is full\n");
                        }
                        else if (front == -1 && rear == -1)
                        {
                              front++;
                              rear++;
                              queue[rear] = data;
                        }
                        else {
                              rear = (rear+1)%MAX;
                              queue[rear] = data;
                        }
                  }

                  int dequeue() {
                        int item;
                        if(front==-1 && rear ==-1) {
                              return -999;
                        }

                        else {
                              item = queue[front];
                              if(front==rear){
                                    front = -1;
                                    rear = -1;
                              }
```

```
            else
                  front = (front+1)%MAX;
            return item;
      }
}

void display() {
      if(front==-1 && rear == -1) {
            printf("Queue is empty\n");
      }
      else {
            printf("Contents of the queue : \n");
            if(front<=rear) {
                  for(int i=front;i<=rear;i++) {
                        printf("%d ",queue[i]);
                  }
                  printf("\n");

            }
            else {
                  for (int i = front; i <= MAX-1; i++) {
                        printf("%d ", queue[i]);
                  }
                  for (int i = 0; i <= rear; i++) {
                        printf("%d ", queue[i]);
                  }
                  printf("\n");
            }
      }
}
int main() {
      int choice;
      int item;
      printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
      printf("Enter your choice : ");
```

```
	scanf("%d",&choice);
	while(choice!=4) {
		switch(choice) {
			case 1: printf("Enter item to be enqueued :
");
				scanf("%d",&item);
				enqueue(item);
				break;

			case 2: item = dequeue();
				if(item == -999) {
					printf("Queue
underflow\n");
				}
				else
					printf("%d has been
dequeued.\n",item);
				break;

			case 3: display();
				break;
		}

printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
		printf("Enter your choice : ");
		scanf("%d",&choice);
	}

}
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./CircularQueue_code
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 1
Enter item to be enqueued : 10
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 1
Enter item to be enqueued : 20
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 1
Enter item to be enqueued : 30
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 3
Contents of the queue :
10 20 30
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 2
10 has been dequeued.
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 2
20 has been dequeued.
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 2
30 has been dequeued.
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 2
Queue underflow
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 3
Queue is empty
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 1
Enter item to be enqueued : 70
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 3
Contents of the queue :
70
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice : 4
```

PROGRAM 5:

WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

CODE:

```
#include<stdio.
h>
                    #include<stdlib.h>


                    struct node {
                          int data;
                          struct node *next;
                    };
                    struct node *head = NULL;



                    int length() {
                          struct node*temp = head;
                          int cnt = 0;
```

```
        while(temp->next!=NULL) {
                cnt++;
                temp = temp->next;
        }
        return cnt+1;
}
void insert_at_end() {
        struct node *newnode, *temp;
        int item;
        newnode = (struct node*)malloc(sizeof(struct node));
        printf("Enter the data : ");
        scanf("%d",&item);

        newnode->data = item;
        if(head == NULL) {
                newnode->next = NULL;
                head = newnode;
                printf("Node Created\n");
        }
        else{
                temp = head;
                while(temp->next!=NULL){
                        temp = temp->next;
                }
                temp->next = newnode;
                newnode->next = NULL;
                printf("Node Created at the end.\n");
        }
}

void insert_at_first() {
        if(head == NULL) {
                insert_at_end();
                return;
        }
```

```
	struct node *newnode;
	int ele;
	printf("Enter the element to be inserted at first
position : ");
	scanf("%d",&ele);
	newnode = (struct node*)malloc(sizeof(struct node));

	newnode->data = ele;
	newnode->next = head;
	head = newnode;
	printf("Element inserted at the first position of the
list.\n");
}

void insert_at_anypos(int pos) {
	if(head==NULL) {
		insert_at_first();
		return;
	}

	if(pos>length()) {
		insert_at_end();
		return;
	}

	struct node *newnode ,*temp;
	temp = head;
	int ele;
	printf("Enter the element to be inserted : ");
	scanf("%d",&ele);
	int jump = 1;
	while(jump<pos-1) {
		temp = temp->next;
		jump++;
	}
	newnode = (struct node*)malloc(sizeof(struct node));
```

```
        newnode->data = ele;
        newnode->next = temp->next;
        temp->next = newnode;
        printf("Element inserted at position %d\n",pos);
}

void display() {
        struct node* ptr = NULL;
        ptr = head;
        if(ptr == NULL)
                printf("No data to print\n");
        else{
                printf("List Contents : \n");
                while(ptr!=NULL) {
                        printf("%d ",ptr->data);
                        ptr = ptr->next;
                }
                printf("\n");
        }
}

int main() {
        int choice;
        int pos;
        printf("SINGLY LINKED LIST\n");
        printf("1.Insert at back\t2.Insert at front\t3.Insert
at any position\t4.Display\t5.Exit\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);

        while(choice!=5) {
                switch(choice) {
                        case 1: insert_at_end();
                                break;
                        case 2: insert_at_first();
                                break;
```

```
                case 3:
                        printf("Enter the postion you
want to insert the new element at : ");
                        scanf("%d",&pos);
                        if(pos==1) {
                              insert_at_first();
                              break;
                        }
                        insert_at_anypos(pos);
                        break;
                case 4: display();
                        break;
          }
          printf("1.Insert at back\t2.Insert at
front\t3.Insert at any position\t4.Display\t5.Exit\n");
          printf("Enter your choice : ");
          scanf("%d",&choice);
     }



     return 0;


}
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./SinglyLinkedList
SINGLY LINKED LIST
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 1
Enter the data : 10
Node Created
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 1
Enter the data : 20
Node Created at the end.
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 1
Enter the data : 30
Node Created at the end.
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 1
Enter the data : 40
Node Created at the end.
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 3
Enter the postion you want to insert the new element at : 4
Enter the element to be inserted : 35
Element inserted at position 4
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 4
List Contents :
10 20 30 35 40
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 2
Enter the element to be inserted at first position : 5
Element inserted at the first position of the list.
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 4
List Contents :
5 10 20 30 35 40
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 3
Enter the postion you want to insert the new element at : 5
Enter the element to be inserted : 37
Element inserted at position 5
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 4
List Contents :
5 10 20 30 37 35 40
1.Insert at back        2.Insert at front       3.Insert at any position        4.Display       5.Exit
Enter your choice : 5
dhruvdubey@Dhruvs-MacBook-Air ds_lab %
```

PROGRAM 6:

WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

CODE:

```
#include<stdio.
h>
                    #include<stdlib.h>

                    struct node {
                          int data;
                          struct node *next;
                    };

                    struct node* head;

                    int length() {
                          struct node*temp = head;
                          int cnt = 0;
```

```
        while(temp->next!=NULL) {
                cnt++;
                temp = temp->next;
        }
        return cnt+1;
}

void insert_at_end() {
        struct node *newnode, *temp;
        int item;
        newnode = (struct node*)malloc(sizeof(struct node));
        printf("Enter the data : ");
        scanf("%d",&item);

        newnode->data = item;
        if(head == NULL) {
                newnode->next = NULL;
                head = newnode;
                printf("Node Created\n");
        }
        else{
                temp = head;
                while(temp->next!=NULL){
                        temp = temp->next;
                }
                temp->next = newnode;
                newnode->next = NULL;
                printf("Node Created at the end.\n");
        }
}

void delete_at_first() {
        if(head==NULL) {
                printf("No elements present to delete\n");
                return;
```

```
        }

        struct node*temp = head;
        head = head->next;

        free(temp);
        printf("Element at first node deleted\n");
}

void delete_at_end() {
        if(head == NULL) {
                printf("No elements present to delete\n");
                return;
        }

        if(head->next == NULL) {
                delete_at_first();
                return;
        }
        struct node*temp = head;
        while(temp->next->next!=NULL) {
                temp = temp->next;
        }
        free(temp->next);
        temp->next = NULL;
        printf("Element at last deleted\n");
}

void delete_at_anypos(int pos) {
        if(head == NULL) {
                printf("No elements present to delete\n");
                return;
        }
        if(pos == 1) {
                delete_at_first();
```

```
            return;
      }

      if(pos > length()) {
            delete_at_end();
            return;
      }
      struct node*prev = head;
      int jump = 1;
      while(jump<pos-1) {
            prev = prev->next;
            jump++;
      }
      struct node*temp = prev->next;
      prev->next = temp->next;
      free(temp);
      printf("Element at %d deleted\n",pos);
}

void display() {
      struct node* ptr = NULL;
      ptr = head;
      if(ptr == NULL)
            printf("No data to print\n");
      else{
            printf("List Contents : \n");
            while(ptr!=NULL) {
                  printf("%d ",ptr->data);
                  ptr = ptr->next;
            }
            printf("\n");
      }
}

int main() {
```

```
	int choice,pos;
	printf("1.Insert at back\n2.Delete at front\n3.Delete
at end\n4.Delete at any position\n5.Display\n6.Exit\n");
	printf("Enter your choice : ");
	scanf("%d",&choice);
	while(choice!=6) {
		if(choice == 1) {
			insert_at_end();
		}
		else if(choice == 2) {
			delete_at_first();
		}
		else if(choice == 3) {
			delete_at_end();
		}
		else if(choice == 4) {
			printf("Enter postion of the element you
want to delete : ");
			scanf("%d",&pos);
			delete_at_anypos(pos);
		}
		else if(choice == 5) {
			display();
		}
		else {
			break;
		}
		printf("Enter your next choice : ");
		scanf("%d",&choice);

	}
	return 0;
}
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./deletionLL
1.Insert at back
2.Delete at front
3.Delete at end
4.Delete at any position
5.Display
6.Exit
Enter your choice : 1
Enter the data : 1
Node Created
Enter your next choice : 1
Enter the data : 2
Node Created at the end.
Enter your next choice : 1
Enter the data : 3
Node Created at the end.
Enter your next choice : 1
Enter the data : 4
Node Created at the end.
Enter your next choice : 1
Enter the data : 5
Node Created at the end.
Enter your next choice : 5
List Contents :
1 2 3 4 5
Enter your next choice : 2
Element at first node deleted
Enter your next choice : 3
Element at last deleted
Enter your next choice : 4
Enter postion of the element you want to delete : 3
Element at 3 deleted
Enter your next choice : 5
List Contents :
2 3
Enter your next choice : 4
Enter postion of the element you want to delete : 1
Element at first node deleted
Enter your next choice : 5
List Contents :
3
Enter your next choice : 6
dhruvdubey@Dhruvs-MacBook-Air ds_lab %
```

PROGRAM 7:

WAP Implement Single Link List with following operations a) a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists.

CODE:

```
#include<stdio.
h>
                    #include<stdlib.h>


                    struct node{
                          int data;
                          struct node*next;
                    };


                    void insertAtEnd(struct node**head,int d){
                          struct node *temp,*n;
                          if(*head == NULL){
                                temp = (struct node*)malloc(sizeof(struct node));
                                temp->data = d;
                                temp->next = NULL;
```

```
            *head = temp;
        }
        else {
            temp = *head;
            //go to the last node
            while(temp->next!=NULL){
                temp = temp->next;
            }
            //adding node at the end
            n = (struct node*)malloc(sizeof(struct node));
            n->data = d;
            n->next = NULL;
            temp->next = n;
        }
}
void reverse(struct node**head) {
        struct node *q,*r,*s;
        q = *head;
        r = NULL;
        if(*head == NULL) {
            printf("Empty LIST\n");
            return;
        }

        while(q!=NULL) {
            s = r;
            r = q;
            q = q->next;
            r->next = s;
        }
        *head = r;
}

void concat(struct node**head1,struct node**head2){
        if(*head1==NULL) {
```

```
            *head1 = *head2;
            return;
      }
      if(*head2==NULL) {
            *head2 = *head1;
            return;
      }
      struct node*temp = *head1;
      while(temp->next!=NULL) {
            temp = temp->next;
      }
      temp->next = *head2;
}

struct node* merger(struct node*a,struct node*b) {
      //base case
      if(a==NULL) {
            return b;
      }
      if(b==NULL) {
            return a;
      }

      struct node*c = NULL;
      //rec case
      if(a->data < b->data) {
            c = a;
            c->next = merger(a->next,b);
      }
      else{
            c = b;
            c->next = merger(a,b->next);
      }
      return c;
}
```

```
struct node* MidPoint(struct node*head){
      if(head == NULL || head->next == NULL){
            return head;
      }

      struct node*fast = head->next;
      struct node*slow = head;

      while(fast != NULL && fast->next != NULL){
            fast = fast->next->next;
            slow = slow->next;
      }
      return slow;
}

struct node* MergeSort(struct node*head){
      if(head == NULL || head->next == NULL) {
            return head;
      }
      //rec case
      //1. Breaking into 2
      struct node* mid = MidPoint(head);
      struct node*a = head;
      struct node*b = mid->next;

      mid->next = NULL;

      //2. rec sort the two parts
      a = MergeSort(a);
      b = MergeSort(b);

      //3. Merging them
      struct node* c = merger(a,b);
      return c;
```

```
}

void display(struct node *head){
	while(head!=NULL){
		printf("%d-->",head->data);
		head = head->next;
	}
	printf("\n");
}



int main()
{
	struct node
*head1=NULL,*head2=NULL,*head3=NULL,*head4=NULL,*ans=NULL;
	int data,n;
	printf("----SORTING----\n");
	printf("Enter the list to be sorted(Enter -1 to stop):
\n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head1,data);
		scanf("%d",&data);
	}
	printf("List before sorting: ");
	display(head1);
	ans = MergeSort(head1);
	printf("List after sorting: ");
	display(ans);


	printf("\n----REVERSE----\n");
	printf("Enter the list to be reversed(Enter -1 to
stop): \n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head2,data);
```

```
            scanf("%d",&data);
        }
        printf("List before reversing: ");
        display(head2);
        reverse(&head2);
        printf("List after reversing: ");
        display(head2);

        printf("\n----CONCATENATION----\n");
        printf("Enter the first list(Enter -1 to stop): \n");
        scanf("%d",&data);
        while(data!=-1) {
            insertAtEnd(&head3,data);
            scanf("%d",&data);
        }
        printf("Enter the second list(Enter -1 to stop): \n");
        scanf("%d",&data);
        while(data!=-1) {
            insertAtEnd(&head4,data);
            scanf("%d",&data);
        }

        printf("First List: ");
        display(head3);
        printf("Second List: ");
        display(head4);
        concat(&head3,&head4);
        printf("Concatenated List: ");
        display(head3);



        return 0;
}
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air desktop % cd ds_lab
dhruvdubey@Dhruvs-MacBook-Air ds_lab % gcc operationsLL.c
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./a.out
----SORTING----
Enter the list to be sorted(Enter -1 to stop):
6 7 3 2 8 -1
List before sorting: 6-->7-->3-->2-->8-->
List after sorting: 2-->3-->6-->7-->8-->

----REVERSE----
Enter the list to be reversed(Enter -1 to stop):
1 2 3 4 5 -1
List before reversing: 1-->2-->3-->4-->5-->
List after reversing: 5-->4-->3-->2-->1-->

----CONCATENATION----
Enter the first list(Enter -1 to stop):
1 2 3 -1
Enter the second list(Enter -1 to stop):
4 5 -1
First List: 1-->2-->3-->
Second List: 4-->5-->
Concatenated List: 1-->2-->3-->4-->5-->
dhruvdubey@Dhruvs-MacBook-Air ds_lab %
```

PROGRAM 8:

WAP to implement Stack & Queues using Linked Representation.

CODE:

```
#include<stdio.h>
#include<stdlib.h>

struct node{
      int data;
      struct node*next;
};
struct node*front;
struct node*rear;

void push(struct node**top,int d) {
      struct node*temp,n;

      temp = (struct node*)malloc(sizeof(struct node));
```

```
	if(temp == NULL) {
		printf("Stack is full\n");
	}

	temp->data = d;
	temp->next = *top;
	*top = temp;
	printf("%d is pushed\n",d);
}

void pop(struct node**top) {
	struct node*temp;

	if(*top==NULL) {
		printf("Stack Underflow\n");
		return;
	}

	temp = *top;
	printf("%d poped\n",temp->data);
	*top = (*top)->next;

	free(temp);
}

void display(struct node* top) {
	if(top == NULL){
		printf("No Elements Present in Stack\n");
		return;
	}

	while(top!=NULL) {
		printf("%d ",top->data);
		top = top->next;
```

```
	}
	printf("\n");
}

void insert(int d) {
	struct node*n;
	n = (struct node*)malloc(sizeof(struct node));
	if(n == NULL){
		printf("Queue Overflow\n");
		return;
	}
	n->data = d;
	if(front==NULL) {
		front = n;
		rear = n;
		front->next = NULL;
		rear->next = NULL;
	}
	else {
		rear->next = n;
		rear = n;
		rear->next = NULL;
	}
	printf("%d is inserted\n",d);
}

void delete() {
	struct node*temp;
	if(front == NULL) {
		printf("Queue Underflow\n");
		return;
	}
	temp = front;
	printf("%d deleted\n",temp->data);
	front = front->next;
```

```
      free(temp);
}

void display_queue() {

    struct node *temp;
    temp = front;
    if(front == NULL)
    {
        printf("\nEmpty queue\n");
    }
    else
    {   printf("\nQueue Elements: \n");
        while(temp != NULL)
        {
            printf("%d ",temp -> data);
            temp = temp -> next;
        }
        printf("\n");
    }
}

int main() {
      struct node*stack = NULL;
      printf("STACK OPERATIONS\n");
      printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
      int choice,item;
      printf("Enter your choice: ");
      scanf("%d",&choice);
      while(choice!=4) {
            switch(choice) {
                  case 1: printf("Enter data to be pushed:
");
                              scanf("%d",&item);
                              push(&stack,item);
                              break;
```

```
                case 2: pop(&stack);
                        break;

                case 3: display(stack);
                        break;
            }
            printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
            printf("Enter your choice: ");
            scanf("%d",&choice);
      }
      printf("End of Stack Operations\n\n");

      printf("QUEUE OPERATIONS\n");
      printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&choice);
      while(choice!=4) {
            switch(choice) {
                case 1: printf("Enter data to be inserted:
");
                        scanf("%d",&item);
                        insert(item);
                        break;

                case 2: delete();
                        break;

                case 3: display_queue();
                        break;
            }
            printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
            printf("Enter your choice: ");
            scanf("%d",&choice);
      }
      printf("End Of Queue Operations\n");
```

```
        return 0;
    }
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./StackAndQueueLL
STACK OPERATIONS
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be pushed: 10
10 is pushed
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be pushed: 20
20 is pushed
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be pushed: 30
30 is pushed
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 3
30 20 10
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 2
30 poped
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 2
20 poped
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 3
10
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 4
End of Stack Operations

QUEUE OPERATIONS
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 10
10 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 20
20 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 30
30 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 3

Queue Elements:
10 20 30
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 2
10 deleted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 2
20 deleted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 3

Queue Elements:
30
```

PROGRAM 9:

WAP Implement doubly link list with primitive operations a) a) Create a doubly linked list. b) Insert a new node to the left of the node. b) c) Delete the node based on a specific value. c) Display the contents of the list

CODE:

```
#include<stdio.
h>
                    #include<stdlib.h>

                    typedef struct Node{
                          int data;
                          struct Node*prev;
                          struct Node*next;
                    }node;

                    void insertAtEnd(node**head,int d) {
                          node *n,*temp=*head;

                          if(*head==NULL) {
                                //add a new node
                                *head = (node*)malloc(sizeof(node));
                                (*head) ->prev = NULL;
                                (*head) ->data = d;
                                (*head) ->next = NULL;
                          }

                          else {
                                //traverse to reach the last node
                                while(temp->next!=NULL) {
                                      temp = temp->next;
                                }
                                //add a new node at the end

                                n = (node*)malloc(sizeof(node));
                                n->data = d;
                                n->next = NULL;
                                n->prev = temp;
```

```
            temp->next = n;
        }
        printf("%d was inserted in the list\n",d);
}

void insertLeft(node**h,int d,int ele) {
        //check if it is the first element
        node*head = *h;
        if(*h==NULL) {
            insertAtEnd(h,d);
            return;
        }
        if(head->data == ele) {
            //code to enter at the start of the list
            node*temp1 = NULL;
            //add a new node
            temp1 = (node*)malloc(sizeof(node));
                //assigning values to temp
            temp1->prev = NULL;
            temp1->data = d;
            temp1->next = *h;
            (*h)->prev = temp1;
            *h = temp1;
            printf("%d was inserted at start\n",d);
            return;
        }

        node*temp = NULL;
        //traverese till we find the element
        while(head!=NULL) {
            if(head->data == ele) {
                head = head->prev;
                temp = (node*)malloc(sizeof(node));
                temp->data = d;
                temp->prev = head;
```

```
                temp->next = head->next;
                temp->next->prev = temp;
                head->next = temp;
                printf("%d was inserted to the left of
%d\n",d,ele);
                return;
          }
          else {
                head = head->next;
          }
     }
     printf("Given element is not present in the list\n");
}

void deleteNode(node**head,int d) {
     node *temp = *head;

     if(*head == NULL) {
          printf("No elements in the list to delete\n");
          return;
     }

     if((*head)->data == d && (*head)->next== NULL) {
          *head = NULL;
          printf("%d was deleted\n", d);
          return;
     }
     while(temp!=NULL) {
          if(temp->data == d) {
                //if node to be deleted is first node
                if(temp == *head) {
                     *head = (*head)->next;
                     (*head)->prev = NULL;
                     free(temp);
                }
                //if last node is to be deleted
```

```
                else if(temp->next == NULL) {
                        temp->prev->next = NULL;
                        free(temp);
                }
                //if node is present somewhere in between
                else {
                        temp->prev->next = temp->next;
                        temp->next->prev = temp->prev;
                        free(temp);
                }
                printf("%d was deleted\n", d);
                return;
            }
            temp = temp->next;
        }
        printf("%d is not present in the list\n",d);
}


void display(node*head) {
        if(head==NULL) {
            printf("Empty List\n");
            return;
        }
        while(head!=NULL) {
            printf("<-%d-> ",head->data);
            head = head->next;
        }
        printf("\n");
}

int main() {
        node*head = NULL;
        int data,pos,opt;
        printf("Insert few elements in the list(Press -1 to
stop) : \n");;
```

```
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head,data);
		scanf("%d",&data);
	}
	printf("Operations on Doubly Linked List\n");
	printf("1.Insert At Left\n2.Delete specified
node\n3.Display\n4.Insert At End\n5.Exit\n");
	printf("Your choice : ");
	scanf("%d",&opt);

	while(opt!=5) {
		switch(opt) {
			case 1: printf("Enter element to be inserted
: ");
				scanf("%d",&data);
				printf("Enter the node : ");
				scanf("%d",&pos);
				insertLeft(&head,data,pos);
				break;

			case 2: printf("Enter the element to be
deleted : ");
				scanf("%d",&data);
				deleteNode(&head,data);
				break;

			case 3: display(head);
				break;

			case 4: printf("Enter data to be inserted at
end: ");
				scanf("%d",&data);
				insertAtEnd(&head,data);
		}
		printf("1.Insert At Left\t2.Delete specified
node\t3.Display\t4.Insert At End\t5.Exit\n");
```

```
                printf("Your choice : ");
                scanf("%d",&opt);
            }
        }
```

OUTPUT:

```
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./DoubleLL
Insert few elements in the list(Press -1 to stop) :
10 12 13 15 -1
10 was inserted in the list
12 was inserted in the list
13 was inserted in the list
15 was inserted in the list
Operations on Doubly Linked List
1.Insert At Left
2.Delete specified node
3.Display
4.Insert At End
5.Exit
Your choice : 1
Enter element to be inserted : 11
Enter the node : 12
11 was inserted to the left of 12
1.Insert At Left        2.Delete specified node 3.Display       4.Insert At End 5.Exit
Your choice : 3
<-10-> <-11-> <-12-> <-13-> <-15->
1.Insert At Left        2.Delete specified node 3.Display       4.Insert At End 5.Exit
Your choice : 1
Enter element to be inserted : 14
Enter the node : 15
14 was inserted to the left of 15
1.Insert At Left        2.Delete specified node 3.Display       4.Insert At End 5.Exit
Your choice : 3
<-10-> <-11-> <-12-> <-13-> <-14-> <-15->
1.Insert At Left        2.Delete specified node 3.Display       4.Insert At End 5.Exit
Your choice : 2
Enter the element to be deleted : 12
12 was deleted
1.Insert At Left        2.Delete specified node 3.Display       4.Insert At End 5.Exit
Your choice : 3
<-10-> <-11-> <-13-> <-14-> <-15->
1.Insert At Left        2.Delete specified node 3.Display       4.Insert At End 5.Exit
Your choice : 5
```

PROGRAM 10:

Write a program

a) To construct a binary Search tree.
b) To traverse the tree using all the methods i.e., in-order, preorder and post order
c) To display the elements in the tree.

CODE:

```
#include<stdio.
h>
                    #include<stdbool.h>
                    #include<stdlib.h>

                    typedef struct binary_node{
                          int data;
                          struct binary_node *left;
                          struct binary_node *right;
                    }node;



                    void insert(node**root,int d) {
                          if(*root == NULL) {
                                (*root) = (node*)malloc(sizeof (node));
                                (*root)->left = NULL;
                                (*root)->data = d;
                                (*root)->right = NULL;
                          }
                          else {
                                if(d<((*root)->data)){
                                      insert(&((*root)->left),d);
                                }
                                else
                                      insert(&((*root)->right),d);
                          }
                    }



                    void inorder(node *root) {
                          if(root == NULL)
                                return;
```

```
	inorder(root->left);
	printf("%d ",root->data);
	inorder(root->right);
}

void preorder(node *root) {
	if(root == NULL) {
		return;
	}

	printf("%d ",root->data);
	preorder(root->left);
	preorder(root->right);
}

void postorder(node *root) {
	if(root == NULL)
		return;

	postorder(root->left);
	postorder(root->right);
	printf("%d ",root->data);
}

bool search(node*root,int key) {
	if(root == NULL)
		return false;

	if(root->data == key) {
		return true;
	}

	else if(key < root->data) {
```

```
            return search(root->left,key);
      }

      else {
            return search(root->right,key);
      }
}

int main() {
      node*root = NULL;
      int choice;
      int d;

      printf("1.Insert in
BST\n2.PreOrder\n3.InOrder\n4.PostOrder\n5.Search\n6.Exit\n"
);
      printf("Your choice: ");
      scanf("%d",&choice);

      while(choice!=6) {
            switch(choice) {
                  case 1: printf("Enter element to be
inserted: ");
                              scanf("%d",&d);
                              insert(&root,d);
                              printf("%d inserted in the
tree\n",d);
                              break;

                  case 2: printf("PreOrder traversal is: \n");
                              preorder(root);
                              printf("\n");
                              break;

                  case 3: printf("Inorder traversal is \n");
                              inorder(root);
                              printf("\n");
```

```
                              break;

                    case 4: printf("PostOrder traversal is \n");
                              postorder(root);
                              printf("\n");
                              break;

                    case 5: printf("Enter element to be searched
");
                              scanf("%d",&d);
                              if(search(root,d)) {
                                   printf("Element found!\n");
                              }
                              else{
                                   printf("Element not
found\n");
                              }
               }
               printf("Your next choice: ");
               scanf("%d",&choice);
          }
          return 0;
}
```

OUTPUT:

```
Last login: Fri Dec 18 21:32:33 on console
dhruvdubey@Dhruvs-MacBook-Air ~ % cd desktop
dhruvdubey@Dhruvs-MacBook-Air desktop % cd ds
dhruvdubey@Dhruvs-MacBook-Air ds_lab % ./BST
1.Insert in BST
2.PreOrder
3.InOrder
4.PostOrder
5.Search
6.Exit
Your choice: 1
Enter element to be inserted: 10
10 inserted in the tree
Your next choice: 1
Enter element to be inserted: 11
11 inserted in the tree
Your next choice: 1
Enter element to be inserted: 9
9 inserted in the tree
Your next choice: 1
Enter element to be inserted: 13
13 inserted in the tree
Your next choice: 1
Enter element to be inserted: 6
6 inserted in the tree
Your next choice: 1
Enter element to be inserted: 17
17 inserted in the tree
Your next choice: 1
Enter element to be inserted: 19
19 inserted in the tree
Your next choice: 1
```