Date.....

t.No. .....

Page No. .....

Name - Ritu Bachle

Branch - CSE (3rd sem)

Subject - Data structure & Algorithm lab
- (307)

Date.....

Expt.No. ..... # INDEX Page No. 2

Teacher's Signature :_____

1. Wap to find avg of n numbers using array.

```
#include <stdio.h>
int main () {
        int array[100], num, sum=0;
        float avg;

    printf("Enter number of elements in this array ");
    scanf("%d", &num);

    printf("Enter ~~number~~ %d element in this array " num);
    for (int i=0; i<num; i++) {
        scanf("%d", &array[i]);
        }
    for (int i=0; i<num; i++) {
        sum = sum + array[i];
        }
    avg = sum / num;
    printf("The avg of array is %f \n", avg);

    return 0;
    }
```

Experiment Name .....

Output:

Enter number of element in this array: 5

Enter 5 number in this array: 1
2
3
4
5

The avg of array is 3.000

Teacher's Signature :_____

2. Wap to create and traverse a singly linked list

```
#include <stdio.h>
#include <stdlib.h>
 Struct Node {
  int data;
 Struct Node *next;
  };
 int main()
  {
 Struct node *head = NULL;
 Struct node *Second = NULL;
 Struct node *third = NULL;

 head=(Struct Node*)malloc(sizeof(struct Node));
Second=(Struct Node*)malloc(sizeof(Struct Node));
third=(Struct Node*) malloc(sizeof(Struct Node));

 head->data = 10;
 head->next = Second;
Second->data = 20;
Second->next = third;
third->data = 30;
third->next = NULL;
printf("%d", third->data);
 return 0;
 }
```

Experiment Name .....

Output

All data is :
10
20
30

Teacher's Signature : _____

Date.....

Expt.No. ..... Page No. 4

3 Wap to insert a new node at the begin position in singly linked list.

```
#include <stdio.h>
#include <stdlib.h>
 void beginsert (int);
 struct node
  {
 int data;
 struct node *next;
  };
 struct node *head;
 void main ()
  {
 int choice, item;
  do
  {
  printf("\n Enter the item which you want to insert?\n");
  scanf("%d", &item);
  beginsert (item);
 printf("\n Press 0 to insert more ? \n");
  scanf ("%d", &choice);
  }
  while ( choice == 0 );
  }
 void beginsert (int item)
```

Teacher's Signature : _____

```
{
struct node *ptr = (struct node *) malloc (sizeof(struct
                    node *));

if (ptr == NULL)

{
printf ("\nOVERFLOW\n");
}
else
{
ptr -> data = item;
ptr -> next = head;
 head = ptr;

printf ("\n Node inserted\n");

   }

}
```

Output

Enter the item which you want to insert?
12
Node inserted
press 0 to insert more?
0
Enter the item which you want to insert?
23
Node inserted
Press 0 to insert more?
2

4 Wap to insert a new node at the particular position in singly linked list

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

struct node *head=NULL;
struct node {
        int data;
    struct node *next;
     };
void ins(int data)
     {
    struct node *temp=(struct node*)malloc(sizeof(struct
      node));
    temp->data = data;
    temp -> next = head;
    head =temp;
     }
void ins_at_pos_n(int data, int position)
     {
  struct node *ptr=(struct node*) malloc(sizeof(struct node));
   ptr -> data = data;
    int i;
  struct node *temp=head;
   if (position==1)
```

```
    {
      ptr -> next = temp;
       head = ptr;
       return;
    }
    for (i=1; i<position-1; i++)
    {
       temp= temp -> next;
    }
    ptr -> next = temp -> next;
    temp -> next = ptr;
}
void display()
{
struct node *temp=head;
printf("\nList: ");
while(temp!=NULL)
{
  printf("\n%d", temp-> data );
   temp= temp -> next;
}
}
  int main()
{
 int i, n, pos, data;
printf("Enter the number of nodes:\n");
scanf ("%d", &n);
```

```
printf("Enter the data for the nodes:\n");
for (i=0; i<n; i++)
 {
  scanf ("%d", &data);
   ins (data);
}
printf ("Enter the data you want to insert in between the
        do node:\n");
scanf ("%d", &data);

printf("Enter the position at which you want to insert the
       node:\n");
scanf ("%d", &pos);
 if (pos>n)

{
  printf (" Enter a valid position: ");
}
 else
 {
   ins-at-pos-n (data, pos);

 }
 display();
 return 0;
```

Experiment Name .....

Output:-

```
Enter the number of nodes:
 5
Enter the data for the nodes:
 44
  5
 22
  6
 95
Enter the data you want to insert in between the nodes:
 100
Enter the position at which you want to insert the nodes:
 4
List:
 44
  5
 22
 100
  6
 95
```

Teacher's Signature :_____

5. Wap to insert a new node at the end position in singly linked list.

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

struct node {
        int data;
    struct node *next;
        }*head;
void createlist (int n);
void insertNodeatEnd (int data);
void displayList ();

int main()
 {
  int n, data;
printf("Enter the total number of nodes: ");
Scanf ("%d", &n);
CreateList (n);

Printf ("\nData in the list \n");
displaylist ();

Printf("\nEnter data to insert at end of the list: ");
Scanf ("%d", &data);
insertNodeAtEnd (data);
```

Teacher's Signature :_____

```
printf("\n Data in the list \n");
displaylist();
return 0;
}
void createlist (int n)
{
struct node *newNode, *temp;
int data, i;
head = (struct node *) malloc (sizeof(struct node));
if (head == NULL)
{
printf("Unable to allocate memory. ");
}
else
{
printf("Enter the data of node 1: ");
scanf("%d", &data);
head -> data = data
head -> next = NULL;
temp = head;
for (i=2; i<=n; i++)
{
newNode = (struct node *) malloc (sizeof(struct node));
if (newNode == NULL)
{
printf("Unable to allocate memory.");
break;
}
```

```
else
{
printf("Enter the data of node %d: ", i);
Scanf ("%d", &data);
newNode -> data = data;
newNode -> next = NULL;
temp -> next = newNode;
temp = temp -> next;
}
Printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
 }
}
void insertNodeAtEnd(int data)
 {
Struct node *newNode, *temp;
newNode = (Struct Node*) malloc (sizeof (Struct node));
if(newNode == NULL)
{
printf ("Unable to allocate memory.");
} else { newNode -> data = data;
         newNode -> next = NULL;
         temp = head;

 While (temp != NULL && temp -> next != NULL)
    temp = tem -> next;
  temp -> next = newNode;
  printf ("DATA INSERTED SUCCESFULL\n");
 }
}
```

```
void displaylist()
{
struct node *temp;
 if (head==NULL)
 {
  printf ("List is empty.");
 }
 else
 {
  temp = head;
  while (temp != NULL)
  {
  printf ("Data = %d \n", temp->data);
   temp = temp -> next;
  }
 }
}
```

Experiment Name .....

Output

```
Data = 10
Data = 20
Data = 30
Enter data to insert at end of the list: 40
DATA INSERTED SUCCESSFULLY
DATA in the list
Data = 10
Data = 20
Data = 30
Data = 40
```

Teacher's Signature :_____

6. Wap to delete a node at the begin position in singly linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node {
        int data;
      struct node *next;
       }*head;
void createlist(int n);
void deleteFirstNode();
void displaylist();
int main()
 {
  int n, choice;
printf("Enter the total number of nodes ");
Scanf("%d", &n);
Createlist (n);
printf("\nData in the list\n");
displayList();
printf("\n Press 1 to delete first node.: ");
Scanf("%d", &choice);
if (choice == 1)
 deleteFirstNode();
Printf("\n Data in the list\n");
 displaylist();
 return 0;
}
```

Teacher's Signature :_____

```
void createlist (int n)
 {
struct node *newNode, *temp;
int data, i;
head = (struct node *) malloc (sizeof (struct node));
if (head == NULL)
{ printf ("Unable to allocate memory "); }
else { printf ("Enter the data of node 1: ");
scanf ("%d ", &data );
  head->data = data
  head-> next = NULL
  temp = head
for (i=2; i<n; i++) {
 newNode = (struct node*) malloc (sizeof (struct node));
if(newNode == NULL) {
printf ("Unable to allocate memory ");
  break;
} else { printf ("Enter the data of node %d ", i);
        scanf (" %d ", &data);
      newNode->data = data;
      newNode -> next = NULL;
      temp -> next = newNode;
      temp = temp -> next;
    }
}
 printf ("SINGLY LINKED LIST CREATED SUCCESSFULL\n");
  }
}
```

```
void deleteFirstNode() {
   struct node *toDelete;
   if (head == NULL) {
    printf("List is already empty");
    }
    else {
         toDelete = head;
         head = head->next;
    printf("\n Data deleted = %d\n", toDelete->data);
     free (toDelete);
    printf("SUCCESSFULLY DELETED FIST NODE FROM LIST\n");
     }
}
 void displaylist() {
   struct node *temp;
   if (head == NULL)
   {
   printf("List is empty");
   }
    else {
         temp = head;
        while (temp != NULL)
         {
        printf("Data = %d\n", temp->data);
        temp = temp->next;
        }
    }
}
```

Experiment Name .....

Output:-

```
Enter the total number of nodes: 5
Enter data 1 : 10
Enter data 2 : 20
Enter data 3 : 30
Enter data 4 : 40
Enter data 5 : 50
SINGLY LINKED LIST CREATED SUCCESSFULLY
DATA in the first
Data = 10
Data = 20
Data = 30
Data = 40
Data = 50
Press 1 to delete first node: 1
Data deleted = 10
SUCCESSFULLY DELETED FIRST NODE FROM LIST
Data in the list
Data = 20
Data = 30
Data = 40
Data = 50
```

Teacher's Signature : _____

7. Wap to delete a node at the particular position in singly linked list.

```
#include<stdio.h>
#include<stdlib.h>
 struct node {
        int data;
   struct node *next;
   } *head;
 void createlist(int n);
 void deletemiddleNode(int position);
 void displaylist();
  int main() {
    int n, position;
 printf("Enter the total number of nodes ");
 scanf("%d", &n);
  Createlist(n);
  printf("\n Data in the list \n");
  displaylist();
 printf("\n Enter the node position you want to delete ");
  scanf("%d", &position);
  deleteMiddleNode(position);
  printf("\n Data in the list \n");
  displaylist();
    return 0;
  }
 void createlist(int n)
 {
```

```
struct node *newNode, *temp;
 int data; i;
head = (struct node *) malloc (sizeof (struct node));
if (head == NULL) {
printf ("Unable to allocate memory");
 } else {
    printf ("Enter the data of node 1: ");
    scanf ("%d", &data);
    head->data = data;
    head -> next = NULL;
    temp = head;
  for (i=2; i<=n; i++) {
  newNode = (struct node *) malloc (sizeof (struct node));
  if (newNode == NULL) {
  printf ("Unable to allocate memory");
    break;
  } else {
    printf ("Enter the data of node %d: ", i);
    scanf ("%d", &data);
    newNode -> data = data;
    newNode -> next = NULL;
    temp->next = newNode;
    temp = temp -> next;
    }
  }
  printf ("Singly linked list created successfully\n");
  }
}
```

```
void deleteMiddleNode(int position)
  {
    int i;
  Struct node *toDelete, *prevNode;
   if (head == NULL)
   { printf("List is already empty."); }
   else {
       toDelete = head;
       prevNode = head;
     for (i=2; i<= position; i++) {
       prevNode = toDelete;
       toDelete = toDelete->next;
     if (toDelete == NULL)
     break;
     }
    if (toDelete != NULL) {
       if (toDelete == head)
          head = head->next;
     prevNode->next = toDelete->next;
      toDelete->next = NULL;
      free (toDelete);
  printf("Successfully deleted from middle of list\n");
    } else {
       printf("Invalid position unable to delete");
     }
   }
}
```

```
void displaylist()
 {
 struct node *temp;
  if (head == NULL)
   {
   printf("List is empty");
   }
  else
   {
   temp = head;
  while (temp != NULL)
    {
    printf("Data = %d\n", temp->data);
    temp = temp -> next;
    }
  }
}
```

Output:

```
Enter the total number of nodes : 4
Enter data 1 : 10
Enter data 2 : 20
Enter data 3 : 30
Enter data 4 : 40
Singly linked list created successfully
Data in the list
  Data = 10
  Data = 20
  Data = 30
  Data = 40
Enter the node position you want to delete : 3
Successfully Deleted from middle of list.
Data in the list
  Data = 10
  Data = 20
  Data = 40
```

8. Wap to delete a node at the end position in singly linked list.

```
#include <stdio.h>
#include <stdlib.h>
void create(int);
void end-delete();
struct node
 {
 int data;
 struct node *next;
 };
struct node *head;
void main()
 {
int choice, item;
do
 {
printf("\n1. Append list \n2. Delete node \n3. Exit \n4. Enter
        your choice.? ");
scanf("%d", &choice);
switch (choice)
 {
case 1:
printf("\n Enter the item \n");
Scanf("%d", &item);
Create (item);
break;
```

Teacher's Signature :

```
Case 2 :
end_delete();
break;
Case 3:
exit(0);
break;
default:
printf("\n Please enter valid choice.\n");
}
} while (choice != 3);
}
void create (int item)
{
struct node *ptr = (struct node *) malloc (sizeof (struct node*));
if(ptr==NULL)
{
printf("\nOVERFLOW\n");
}
else {
ptr -> data = item;
ptr -> next = head;
head = ptr;
printf("\n Node inserted \n");
}
}
void end_delete()
{
```

```
struct node *ptr, *ptr1;
 if (head == NULL)
  {
 printf("\n list is empty");
 }
 else if (head -> next == NULL)
  {
  head = NULL;
  free (head);
 printf("\n Only node of the list deleted...");
  }
 else {
   ptr = head;
 while (ptr -> next != NULL)
   {
   ptr1 = ptr;
   ptr = ptr -> next;
   }
  ptr1 -> next = NULL;
   free (ptr);

  printf("\n Deleted Node from the last");

   }
}
```

Teacher's Signature :_____

Experiment Name .....

Output.

1. Append list
2. Delete node
3. Exite
4. Enter your choice? 1

Enter the item
12

Node inserted

1. Append list
2. Delete node
3. Exit
4. Enter your choice? 2

Only node of the list deleted...

Teacher's Signature :_____

9. Wap to implementation of binary search in C language.

```
#include <stdio.h>
void binary_search();
 int a[50], n, item, loc, beg, mid, end, i;

 main()
 {
 printf("\n Enter size of an array ");
 Scanf("%d", &n);
 printf("\n Enter elements of an array in sorted form\n");
 for (i=0; i<n; i++)
   Scanf("%d", &a[i]);
 printf("\n Enter ITEM to be searched ");
 Scanf("%d", &item);

 binary_search();
 getch();
}
void binary_search()
 {
  beg = 0
  end = n-1;
  mid = (beg + end)/2;
  while ((beg <= end) && (a[mid] != item))
```

```
{
 if (item < a[mid])
   end = mid - 1;
 else
   beg = mid + 1;
  mid = (beg + end)/2;
}

if (a[mid] == item)

printf("\n\nITEM found at location %d", mid+1);
else
  printf("\n\nITEM doesn't exist");

}
```

Experiment Name .....

Output

Enter size of an array : 6

Enter elements of an array in sorted form:

10 20 30 40 50 60

Enter ITEM to be searched: 30

ITEM found at location 3

Teacher's Signature : _____

10. Wap to implementation of stack in C language.

```
#include <stdio.h>
int MAXSIZE = 8;
int stack [8];
int top = -1;
int isempty() {
   if (top == -1)
    return 1;
   else
    return 0;
}
int isfull() {
  if (top == MAXSIZE)
   return 1;
  else
   return 0;
}
int peek() {
   return stack [top];
}
int pop() {
   int data;
  if (!isempty()) {
   data = stack [top];
   top = top - 1;
   return data;
```

```
    } else {
        printf("Could not retrieve data, Stack is empty\n");
    }
}
int push(int data){
    if(!isfull()){
        top = top + 1;
    stack[top] = data;
    }else {
    printf("Could not insert data, Stack is full.\n");
    }
}
int main(){
    push(3);
    push(5);
    push(9);
    push(1);
    push(12);
    push(15);
printf("Element at top of the stack: %d\n", peek());
printf("Elements:\n");
while(!isempty()){
    int data = pop();
    printf("%d\n", data); }
printf("Stack full: %s\n", isfull()?"true":"false");
printf("Stack empty: %s\n", isempty()?"true":"false");
    return 0;
}
```

Teacher's Signature :

Experiment Name .....

Output

Element at top of the stack : 15

Elements :

15
12
1
9
5
3

Stack full : false
Stack empty: true.

Teacher's Signature : _____

11 Wap to implementation of Queue in C language.

```
#include <stdio.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = -1;
int front = -1;
main()
{
int choice;
while(1)
{
printf("1. Insert element to queue.\n");
printf("2. Delete element from queue.\n");
printf("3. Display all elements of queue.\n");
printf("4. Quit\n");
printf("%d", &choice);
Switch(choice)
{
Case 1:
insert();
break;
Case 2:
delete();
break;
```

```
Case 3:
display();
break;
Case 4;
exit(1);
default:
printf("Wrong choice \n");
  }
 }
}
void insert()
 {
   int add_item;
   if (rear == MAX-1)
   printf("Queue Overflow \n");
    else {
      if (front == -1)
      front = 0;
    printf("Insert the element in queue ");
     Scanf("%d", &add_item);
     rear = rear + 1;
     queue_array[rear] = add_item;
   }
}
void delete()
 {
   if (front == -1 || front > rear)
```

```
{
  printf ("Queue Underflow \n");
  return;
}
else
  {
  printf ("Element deleted from queue is %d \n",
            queue_array[front]);
  front = front + 1;
  }
}
void display()
{
  int i;
  if (front == -1)
    printf ("Queue is empty \n");
  else {
    printf ("Queue is: \n");
    for (i=front; i<= rear; i++)
      printf ("%d", queue_array[i]);
      printf ("\n");

  }
}
```

Output

1. Insert element to queue
2. Deleted element from queue
3. Display all elements of queue
4. Quit

Enter your choice : 1
Insert the element in queue : 10
1. Insert element to queue
2. Delete element from queue
3. Display all element of queue
4. Quit
Enter your choice : 1
Insert element in queue : 15
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice : 1
Insert the element in queue : 20
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice : 1
Insert the element in queue : 30
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice : 2
Element deleted from queue is : 10
1. Insert
2. Delete
3. Display
4. Quit

12. Wap to implementation of merge sort in C language.

```
#include<stdio.h>
#define max 10
int a[11] = {10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0};
int b[10];
void merging(int low, int mid, int high){
    int l1, l2, i;
  for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++){
    if(a[l1] <= a[l2]
        b[i] = a[l1++];
    else
      b[i] = a[l2++];
  } while(l1 <= mid)
    b[i++] = a[l1++];
   while(l2 <= high)
    b[i++] = a[l2++];
   for(i = low; i <= high; i++)
    a[i] = b[i];
}
void sort(int low, int high){
    int mid;
  if(low < high){
    mid = (low + high)/2;
    sort(low, mid);
   sort(mid+1, high);
   merging(low, mid, high);
  }
```

```
    else {
      return;
    }
}
int main(){
    int i;
    printf("List before sorting.\n");
    for(i=0; i<=max; i++)
    printf("%d", a[i]);
    bsort(0, max);
    printf("\nList after sorting.\n");
    for(i=0; i<=max; i++)
    printf("%d", a[i]);
    }
```

Output:

List before sorting.

10 14 19 26 27 31 33 35 42 44 0

List after sorting.

0 10 14 19 26 27 31 33 35 42 44

19. Wap to implementation of quick sort in c language

```
#include <stdio.h>
int partition (int a[], int start, int end) {
 int pivot = a[end];
 int i = (start-1);
 for (int j= start; j<= end-1; j++) {
   if (a[j]< pivot) {
    i++;
    int t = a[i];
    a[i] = a[j];
    a[j] = t;
   }
 }
   int t = a[i+1];
   a[i+1]= a[end];
   a[end] = t;
   return (i+1);
  }
void quick (int a[], int start, int end)
   {
    if (start < end)
    {
  int p= partition(a, start, end);
  quick (a, start, p-1);
  quick (a, p+1, end);
   }
}
```

Experiment Name .....

```
void printArr(int a[], int n) {
    int i;
    for(i=0; i<n; i++)
    printf("%d ", a[i]);
    }
    int main() {
     int a[] = {24, 9, 29, 14, 19, 27};
     int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting, array elements are -\n");
    printArr(a, n);
     quick(a, 0, n-1);
    printf("\n After sorting array elements are-\n");
    printArr(a, n);
     return 0;
    }
```

Output

Before sorting array elements are -
24 9 29 14 19 27
After sorting array elements are -
9 14 19 24 27 29

Teacher's Signature : _____

14. Wap to implementation of selection sort in C language

```
#include <stdio.h>
int main() {
int arr[10] = { 6, 12, 0, 18, 11, 99, 55, 45, 34, 2};
int n = 10;
int i, j, position, swap;
for( i = 0; i < (n-1); i++) {
 position = i;
 for(j = i+1; j < n; j++) {
  if (arr[position] > arr[j])
   position = j;
 }
 if (position != i) {
  swap = arr[i];
 arr[i] = arr[position];
 arr[position] = swap;
 }
}
 for (i=0; i<n; i++)
  printf ("%d\t", arr[i]);
  return 0;
}
```

Experiment Name .....

Output:

0 2 6 11 12 18 34 35 45 55 99

Teacher's Signature :_____