PAGE NO.:
DATE: / /

Lab file

Bhoomi Motiani
Sem 2 (1st year)
B.Tech CSE.
A20405220004.

① Write a C program to reverse a given number & also calculate the number of digits in the number.

→
```
#include <stdio.h>
int main() {
    int num, r, reverse=0;

    printf("Enter any number :");
    scanf("%d", &num);

    while (num) {
        r = num%.10;
        reverse = reverse * 10+r;
        num = num/10;
    }
    printf("Reversed of number: %d ", reverse);
    return 0;
}
```

Sample output:
Enter any number : 12
Reversed of number : 21

Bhoomi Motiani

PAGE NO.:
DATE: / /

(2) Write a C program to find the minimum/maximum number in a given array.

→

```
#include<stdio.h>
int main()
{
   int a[1000], i, n, min, max;

   printf("Enter size of the array: ");
   scanf("%d", &n);

   printf("Enter elements in array:");
   for(i=0; i<n; i++)

   {
      scanf("%d", &a[i]);
   }

   min = max = a[0];
    for(i=1; i<n; i++)

    {
       if (min > a[i])
          min = a[i];
          if (max < a[i])
             max = a[i];
       }
      printf("minimum of array is: %d", min);
       printf("\nmaximum of array is: %d max);

   return 0;
}
```

PAGE NO.:
DATE: / /

Q-3 Write a C program to input an array of size 10 & print & average of all array elements.

→

```
#include <stdio.h>
int main()
{
  int i, num=10;
  float total = 0.0, average;
  int array[num];
  printf("Enter %d numbers (-ve +ve or zero)\n", num);

  for i=0; i<num; i++)
  {
   scanf("%d", &array[i]);
  }
  printf("Input array elements\n");
  for(i=0; i<num, i++);
  {
   printf("%+3d\n", array[i]);
  }
   for(i=0; i<num; i++)
  {

  total += array[i]; / this means total = total + array[i]; /
}
```

```
average = total / num;
printf("\n Sum of all numbers = %.2f \n", total);
printf("\n Average of all input numbers =
        %.2f \n", average);
}
```

Output if entered {1, 2, 3, ... = 10}
as input

Enter 10 numbers (-ve, +ve & 0)

1
2
3
4
5
6
7
8
9
10

Input array elements
+1
+2
+3
+4
+5
+6
+7
+8

+9
+10

Sum of all numbers = 55.00

Average of all input numbers = 5.50

PAGE NO.:
DATE: / /

(4) Write a C program that input n numbers & print the reverse of the array.

→
```
#include <stdio.h>
int main()
{
    int arr[] = {1, 2, 3, 4, 5};

    int length = sizeof(arr)/sizeof arr[0];

    printf("Original array: \n");

    for int i = 0; i < length; i++) {
     printf("%d", arr[i]);

    } }

    printf("\n");

printf (array in reverse order: \n");

for int i = length - 1; i >= 0; i--){

    printf("%d", arr[i]);

    }
return 0

}
```

PAGE NO.:
DATE: / /

(5) Write a C program to implement the concepts of dynamic array using malloc / calloc & free function. And print sum & average of all array functions.

→
```
#include <stdio.h>
int main()
{
  int sum = 0 avg, i;
  int how_many;
  int* p;

  Printf ("Enter numbers of Subjects whose marks you want to input:");
  Scanf ("%d" &how_many);

  P = (int*) malloc (how many * 4);

  if(p == NULL)

  {
    printf("Out of memory !\n");
    printf("Press any key to close");
    exit(0);
  }

  for(i = 0; i < how_many, i++)

  {
    printf ("Enter marks of %d th subject :" i+1);
    Scanf("%d", &p[i]);
```

```
        sum = (sum + p[i]);
    }
    avg = (sum / how_many);
    printf("sum = %d \n\n", sum);
    printf("Average = %d \n\n", avg);
    free(p);
    printf("press any key to close");
    getch();
}
```

(6) Write a C program to implement the concept of passing array to function using pointer

→

```
#include <stdio.h>
void display (int *num)
{
    printf("%d", *num);
}

int main()
{
    int arr[7] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
    for (int i=0; i<10; i++)
    {
        disp (&arr[i]);
    }
} return 0
```

PAGE NO.:
DATE: / /

(7) Write a C program implement the concepts of simple structure with one structure variable.

→

```
#include<stdio.h>

struct StudentData {
char *stu_name;
int stu_id;
int stu_age;
};

int main()
{
    struct StudentData student;

    student.stu_name = "steve";
    student.stu_id = 1234
    student.stu_age = 30

printf("Student name is: %s", student.stu_name);
printf("student Id is : %d", student.stu_id);
printf("student age is : %d", student.stu_age);

return 0;
}
```

name of structure is student data

: student is the variable

Output

student name is : Steve
student id is : 1234
student age is: 30

(8) Write a C program to store the records of 5 books & print it using array of structure.

```
#include <stdio.h>

struct Book
{
   int number;
   float rating;
};
int main()
{
  int i=0, n=5;
  struct book book[n];
  book[0].number = 1;
  book[0].rating = 5.5;

  book[1].number = 2;
  book[1].rating = 7.5;

  book[2].number = 3;
  book[2].rating = 10;

  book[3].number = 4;
  book[3].rating = 9.5;

  book[4].number = 5
  book[4].rating = 8.5
  printf("Book records:\n\n");
  for(i=0; i<n; i++){
  printf("Name = %d\n", book[i].number);
  printf("Rating = %.2f\n" book[i].rating);
}
return 0
```

PAGE NO.:
DATE: / /

(9) Write a C program to implement the concepts of for passing an array of structure to function using pointer.

→
```
#include<stdio.h>

struct student
{
    int id;
    char name[20];
    float percentage;
};

void func(struct student *record);

int main()
{
  struct student record;

    record.id=1
    strcpy(record.name, "Bhoomi");
    record.percentage = 92.5;

    func(&record);
     return 0;
}

Void func(struct student *record)
 {
   printf("Id is: %d\n" record->id);
   printf(" Name is %s\n" record->name);
   printf("Percentage is: %f\n", record->percentage);
} return 0
```

Output
Id is : 1
Name is : Bhoomi
Percentage is : 92.500000

(10) Write a C program create & display singly linked list.

→
```
#include <stdio.h>

struct node
{
   int num;
   struct node * nextptr;
} * stnode;

void create node list (int n);
void ( display list ();

int main ()
{
    int n;
    printf ("\n\n Linked list : To create
     and * display singly linked lis: \n");

    printf ("Input the number of nodes: ");
    scanf ("%d", &n);

    create Node list (n);
    printf ("\n Data entered in the list : \n");
     display list ();
} return 0
```

```
void CreateNodeList(int n)
{
  struct node *fnNode, *tmp;
  int num, i;
 st node = (struct node *) malloc(sizeof(struct
                                    node));

  if (st node == NULL)

  {
    printf("\n Memory can not be allocated.")
  }

  else
  {
    printf("Input data for node 1: ");
   Scanf("%d", &num);

   st node -> num = num;
   st node -> nextptr = NULL;
   tmp = st node;

   for(i=2; i<=n; i++)
   {
     fnNode = (struct node*) malloc(sizeof
                                    (struct node));
     if (fnNode == NULL)

   }}
    else
  {
     printf("Input data for node! ", i);
     Scanf("%d", &num);
```

```
fn Node -> num = num
fn Node -> nextptr = NULL;

tmp -> nextptr = fn Node;
tmp = tmp -> nextptr
}
}
}
}
```

~~void disp~~

```
void display list ()
{
struct node *tmp;
if (stnode == NULL)
{
printf ("List is empty.");
}
else
{
tmp = stnode;
while (tmp != NULL)
{
printf ("Data = %d \n", tmp->num);
tmp = tmp -> next ptr;
}
}
}
```

PAGE NO.:
DATE: / /

Output

```
Input the number of nodes : 3
Input data for node 1 : 5
Input data for node 2: 6
Input data for node 3 : 7

Data entered in the list
Data = 5
Data = 6
Data = 7
```

(12) Write a C program to delete an element from existing linked list using all three way.

→
```
#include <stdio.h>

// creating node with data and a pointer named node
{
int data;
struct node *next;
} *head;
void creatlist(int n);
void deletion_beginning();
void displaylist();

int main
{
int n, data, pos;

printf("\Enter the total number of nodes:");
Scanf("%d", &n);
if(n==0)
{
  printf("Empty list\n");
  exit(0);
}
  else
{
  createlist(n);
}
```

```
printf("\n The list is \n");
display list();
deletion - beginning();
printf("\n\n After deleting the first node,
 the list is \n");
display list();
return 0.
}

Void createlist (int n)
{
  struct node * newnode * temp;
  int data, i;
head = (struct node *) malloc (sizeof(struct node));

// when the list is empty

if (head == NULL)
{
  printf ("Unable to allocate memory.");
}
  else
{
  printf ("\n Enter the data of node 1:");
  Scanf ("%d", &data);
  head -> data = data;
  head -> next = NULL;
  temp = head;
  for (i = 2; i <= n; i++)
  {

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

```
if (newNode == NULL)
{
  printf ("unable to allocate memory.");
break;
}
 else
{
printf ("\n Enter the data of node %d : ", i);
scanf ("%d", &data);
new node -> data = data;
new node -> next = null;
 temp -> next = newNode;
 temp = temp -> next;
 }
 }
 }
}

/* function to delete the first node */ void
 deletion_beginning ()
{
  // Empty list
 if (head == NULL)
 printf ("\n The list is empty \n");
 struct node * temp;
 temp = head; // make temp as head node.
 head = head -> next; // shift the head node
 free (temp); // Delete the temporary node.
}
  Void displaylist()
{
```

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

Output:

```
Enter the total no. of nodes: 4
Enter the data of node 1: 5
Enter the data of node : 2: 10
Enter the data of node : 3: 15
Enter the data of node : 4: 20

The list is
 5  10  15  20
After deleting the first node, the list is
 10  15  20
```

(13) Write a C program to search an element from the linked list.

→

```
#include<studio.h>
struct node
{
  int num;
  struct node *nextptr;
}
  st node, *ennode;

int Find Element (int);
void main()
 {
   int n, i, Find Elem, Find Plc;
    st node.nextptr = NULL;
    ennode = & st node
 }

int find element (int Find)
printf("\n\n Linked list: Seasch an element
        in a Sngly linked list: \n");

printf("Input the number of node:");
 scanf("%d", &n);
 printf("\n");
for(i=0; i<n; i++)
{
    ennode -> next ptr = (struct node *) mallo
                         (Size of (struct node))
    printf("Input data for node %d: ", i+1);
    scanf("%d", & ennode -> num);
```

```
    ennode = ennode -> nextptr;
}
ennode -> nextptr = NULL;
printf ("\n Data entered in the list are: \n");

ennode = &stnode;
while (ennode -> nextptr != NULL)
{
    printf ("Data = %d \n", ennode -> num);
    ennode = ennode -> next ptr;
}

printf ("\n");
printf ("Input the element to be searched:");
scanf ("%d", &find elem);
Find plc = Find element (Find elem);
    if (Find Plc <= n)

printf ("Element found at node %d \n\n", Find
                                        Plc);
else
printf ("This element does not exists is linked
list \n\n");
}
int Find element (int find elem)
{
    int ctr = 1;
    ennode = &st node;
    while (ennod -> nextptr != NULL)
    {
```

```
    if (enmode -> num == Findelem)
        break;
    else
        ctr++;
        enode = enmode -> nextptr;
    }
    return ctr;
}
```

Output:

```
Input the number of nodes : 3

Input data for node 1 : 2
Input data for node 2 : 5
Input data for node : 3 : 8

Data entered in the list are:
Data = 2
Data = 5
Data = 8

Input the element to be searched : 5
Element found at node : 2
```

PAGE NO.:
DATE: / /

(14) Write a C program create & display circular linked list.

→

```
# include <stdio.h>

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

  struct node * head = NULL;
  struct node * tail = NULL;

  void add (int data) {
  struct node *NewNode = (struct node*) malloc (sizeof
                                  (struct node));

  new node -> data = data.

  if (head == NULL) {

  head = NewNode
  tail = new Node;
   newnode -> next = head;
  }
  else
 {
  tail = newNode;
 tail -> next = head;
}
}
```

```
void display( ){
struct node* current = head.
 if (head == NULL){
  printf("List is empty");
 }

 else {
   printf ("Nodes of the circular linked list:
                                        \n");

 do {

  printf ("%d", current -> data);
     current = current -> next;
  }

 while (current != head);
   printf ("\n");
 }

}
int main ()

 add (1);
 add (2);
 add(3);
 add (4);

 display ();

 return 0;
}
```

Output: Nodes of the circular linked list : 1 2 3 4

PAGE NO.:
DATE: / /

(15) Write a program to search an element from circular linked list

→

```
#include <stdio.h>

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

struct node *head = NULL;
struct node * tail = NULL;

void add (int data){
struct node * New node = (struct node*) malloc
                         (sizeof (struct node));

new node -> data = data.

if (head == NULL) {

    head = new Node;
    tail = new Node

new node -> next = head.
}
else
{
    tail -> next = new node;

    tail = new node;
```

```
    tail -> next = head
  }
}
void search (int element) {

struct node * current = head;

int i = 1;
bool flag = false
if (head == NULL) {

printf("List is empty");

}
else {
 do {
     if (current -> data == element) {

  flag = true;

   break;
  }

  current = current -> next;
  i++;
}
while (current != head);

if (flag)
```

```
        printf ("Element is present in the list at
                the position: %d", i);

    else
        printf ("\n Element is not present in the
                list")
    }
}

int main()
{
 add(1);
 add(2);
 add(3);
 add(4);

search(2);

search(7);

return 0;
}
```

Output: Element is present in the list at the position: 2
Element is not present in the list.

PAGE NO.:
DATE: / /

(16) Write a program to merge a linked list into another linked list at Alternate positions.

→
```
#include<stdio.h>

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

  void push (struct node** head_ref, int
       new_data).

  struct Node* new_node = (struct Node*)malloc
                  (sizeof(struct node));

  new Node->data = new_data;
  new_node->next = (*head_ref);
  (*head_ref) = new_node;

}
  void printlist (struct Node * head)
  {
   struct Node* temp = head;
   while (temp != NULL)
     {
       printf("%d"; temp->data)
        temp = temp->Next;
     }
```

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

void merge(struct Node *p, struct node **q)
{
  struct node *p_curr = p, *q_curr = *q;
  struct Node *p_next, *q_next;

while (p_curr != NULL && q_curr != NULL)
  {

    p_next = p_curr -> next;
    q_next = q_curr -> next;


    q_curr -> next = p_curr;
    p_curr -> next = q_curr;


    p_curr = p_next;
    q_curr = q_next;

  }

  *q = q_curr;
}

int main()
{
  struct node *p = NULL, *q = NULL;
```

```
push(&p, 3);
push(&p, 2);
push(&p, 1);

printf("First linked list:\n");
printlist(p);

push(&q, 8);
push(&q, 7);
push(&q, 6);
push(&q, 5);
push(&q, 4);

printf("Second linked list:\n");
printlist(q);

merge(p, &q);

printf("modified first linked list:\n");
printlist(p);

printf("modified second linked list:\n");
printlist(q);

getchar();
return 0;
```

Output: First linked list: 1 2 3
Second linked list: 4 5 6 7 8
Modified first linked list: 1 4 2 5 3 6
Modified second linked list: 7 8

PAGE NO.:
DATE: / /

(14) Write a program to sort all singly linked list elements & print it using bubble Sorting.

→ #include <stdio.h>.

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

struct node * head, * tail = NULL;

Void add Node (int data) {

struct node * newNode = (struct node*) malloc
                        (size of (struct node));

  newNode -> data = data;
  new Node -> next = NULL;

if (head == NULL){

  head = newNode
  tail = newNode
}
else {
tail -> next
tail = newN. = new Node,
tail = ne w Node
}
}
```

```
void sortList(){
struct node* current = head, *index = NULL;
int temp;
if(head==NULL){
return;
}
else{
while(current!=NULL){
index = current->next;
while(index!=NULL){
if(current->data > index->data){
{
temp = current->data;
current->data = index->data;
index->data = temp;
}
index = index->next;
}
current = current->next;
}
}
}
```

```
void display(){

Struct node * current = head;
 If (head == NULL) {

printf ("List is empty \n");

return ;

}
while (current != NULL){

printf ("%d ", current->data);

        current = current -> next;
}

printf ("\n");

}

int main( )

{
 addNode (9);
 add Node (7);
 add Node (2);
 add Node (5);
 add Node (4);
```

```
printf ("Original list: \n");
display ();

sortlist ();

printf ("sorted list : \n");
display ();

return 0;
}
```

Output : Original list:
9 7 2 5 4
Sorted list:
2 4 5 7 9.

(18) Write a program to reverse a singly linked list.

→
```
#include <stdio.h>

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

Void Create (struct node**);
Void reverse (Struct node**);
Void release (Struct node**);
Void display (struct node*);
```

```
int main()
{
  Struct node *p = NULL
  int main n;

  printf("Enter data into the list \n");
   create (&p);

  printf("Displaying the nodes in the list s:\n");
   display (p);

  printf (Reversing the list ...\n");
   reverse (&p);

  printf ("Displaying the reversed list:\n");
   display (p);
  release (&p);

  Return 0;

}

Void reverse (Struct node **head)
{

  Struct node *p, *q, *r;

  p = q = r = *head;
  p = q -> next -> next;
  q = q -> next;
  r -> next = NULL;
```

```
q->next=r;
while (p!=NULL)
{
    r=q;
    q=p;
    p=p->next;
    q->next=r;
}
*head=q;
}
Void create(struct node **head)
{
    int c,ch;
    struct node *temp,*rear;
    do
    {
        printf("Enter number:");
        scanf("%d",&c);
        temp=(struct node*)malloc(sizeof(struct
                                  node));
        temp->num=c;
        temp->next=NULL;
        if(*head==NULL)
```

```
    {
        *head = temp;
    }
    else
    {
        rear -> next = temp;
    }

    rear = temp;
    printf ("Do you wish to continue [1/0]: ");
    scanf ("%d", &ch);
  }
  while (ch != 0);
  printf ("\n");
}

void display (struct node *p)
{
  while (p != NULL)
  {
    printf ("%d\t", p -> num);
    p = p -> next;
  }

  printf ("\n");
}

void release (struct node **head)
{
  struct node * temp = *head;
```

```
*head = (*head)->next;
while((*head)!=NULL
{
    free(temp);
    temp = *head;
    (*head)=(*head)->next;
}
}
```

Output:

```
Enter number : 1
Do you wish to continue [1/0] : 1
Enter number : 2
Do you wish to continue [1/0] : 1
Enter number 3
Do you wish to continue [1/0] : 1
Enter number 4
Do you wish to continue [1/0] : 1
Enter number 5
Do you wish to continue [1/0] : 0

Displaying the nodes in the list :

1    2    3    4    5
Reversing the list
Displaying the reversed list :

5    4    3    2    1
```

```
int main(){
int stackSize = 4;
char stack[stackSize];

int top = -1;
push('a' stack & top, stackSize);
printf("Element on top: %c\n", stack[top]);
push('b' stack & top, stackSize);
printf("Element on top: %c\n", stack[top]);
pop(stack, & top, stackSize);
printf("Top: %d\n", top);
pop(stack, & top, stackSize);
return 0
}
```

PAGE NO.:
DATE: / /

(19) Write a C program to implement the concepts of stack using Array.

```
#include <stdio.h>
void push (char element, char stack[], int* top, int
stack size) {
if (*top == -1){
stack[stack size -1] = element;
*top = stack size -1;
}
else if (*top == 0) {
printf("The stack is already full.\n");
}
else{
stack[(*top) -1] = element;
(*top) - -;
}
}
void pop(char stack[], int* top, int stack size) {
if (*top == -1){
printf("The stack is empty.\n");
}
else
printf("Element popped %c\n", stack[(*top)]);

if (*top) == stackSize - 1){
(*top) = -1;
}
else{
(*top)++;
}
}
}
```

PAGE NO.:
DATE: / /

(20) Write a C program to implement the concepts of stack using linked list.

→
```
#include <stdio.h>

struct Node
{
int data;
struct Node * Next
}* top = NULL;

void push (int);
void pop();
void display();

int main()
{
int choice, value;

printf("\n Implementing stacks using linked list\n");

while(1){
printf("1. push\n2. Pop\n3. Display\n4. Exit\n");
printf("Enter your choice");
scanf("%d" & choice);
switch(choice)
{
case 1: printf("\n Enter the value to insert:");
scanf("%d", & value);
push(value);
break;
```

```
case 2 : pop()
break;

case 3 : display()
break;

case 4: exit(0)
break;

default printf ("\n Invalid choice \n");

}}}

Void push(int Value)
{
struct Node * Newnode;
new node = (struct Node*) malloc (Size of(
                                  struct Node));
 if(top == NOLL)
newNode -> NULL

else
 newNode -> next = top;
 top = new Node ;
 printf ("Node is inserted(n) n");
 }

Void pop()
{
 if (top == NULL)
printf ("\n empty stack \n");
```

```
else {
struct Node * temp = top;
printf("\n Popped element: %d", temp -> data);
 printf("\n")
top = temp -> next;
free (temp);
}}

Void display()
 {
  if (top == NULL)

printf("\n Empty stack \n");

else
{
 printf("The stack is \n");
 struct Node *temp = top;
 while (temp -> next != NULL) {
printf ("%d --> ", temp -> data);
temp = temp -> next;
}

printf ("%d --> NULL \n\n", temp -> data

}}.
```