FREEMIND

Date

Page 1

Assignment - I

1) find the factorial of a Number using User defined function.

Code ⇒

```
#include<stdio.h>
    int factorial(int n);
    int main(){
      int n;
      printf("Enter the Number: ");
      scanf("%d", &n);
      factorial(n);
    return 0;
    }
    int factorial(int n){
      int i, f=1;
    for(i=1; i<=n; i++){
      f=f*i;
    }
    printf("Factorial of %d = %d", n, f);
    }
```

Dry Run :-

n = 4	i = 1	f = 1	
	i = 1	f = 1*1	f = 1
	i = 2	f = 1*2	f = 2
	i = 3	f = 2*3	f = 6
	i = 4	f = 6*4	f = 24

Output :-

Enter the Number: 4

Factorial of 4 = 24

2) Print the fibonacci series using User defined function.

Code ⇒

```
#include <stdio.h>
int fibonacci (int n);
int main(){
  int n;
  printf("Enter the Number: ");
  Scanf("%d", &n);
  fibonacci (n);
 return 0;
}

int fibonacci (int n) {
  int i, t1=0, t2= 1, ne=t1+t2;
 printf("fibonacci Series %d, %d, ", t1,t2);
 for (i=3; i<=n; i++){
    printf("%d, ", ne);
  t1 = t2;
  t2 = ne;
  ne = t1+t2;
  }
}
```

Output -

```
Enter the Number : 5
fibonacci Series  0, 1,
                  1, 2, 3, 5,
```

Dry Run

n=5	i=3	t1=0	t2=1	ne=1
		t1=1	t2 1	ne = 2
	i=4	t1=1	t2=2	ne=3
	i=5	t1=2	t2=3	ne=5

3.) Reverse a given Number using User defined function.

Code ⇒

```
#include<stdio.h>
int Reverse(int n);
int main(){
    int n;
    printf("Enter the Number:");
    scanf("%d",&n);
    Reverse(n);
    return 0;
}
int Reverse(int n){
    int i,r;
    for(i=0;i<=n;i++){
    r=n%10;
    n=n/10;
    printf("%d",r);
}
}
```

Output -

Enter the Number:
345

543

Dry Run -

n = 345

i=0	r=n%10	n=n/10
,	r=5	n=34
i=1	r=4	n=3
i=2	r=3	n=0
i=3 x		

4) Find the largest element of the array using user defined function.

Code ⇒

```
#include<stdio.h>
int largest(int ar[], int size);
int main(){
    int ar[100], size, i;
    printf("Enter the Size of Array : ");
    Scanf("%d", &size);
    for(i=0; i<size; i++){
        printf("Enter the Element of Array : ");
        Scanf("%d", &ar[i]);
    } largest(ar, size); }
    return 0;
}
int largest(int ar[], int size){
    int L=a[0];
    for(int i=0; i<size; i++){
if(L<a[i])  if(a[0]<a[i]){
        L a[0]=a[i];
    }
}
    printf("largest Element in Array : %d", L a[0]);
}
```

Output :-

Enter the size of Array : 5
Enter the Element of Array : 1
Enter the Element of Array : 2
Enter the Element of Array : 3
Enter the Element of Array : 4
Enter the Element of Array : 5

largest Element in Array : 5

Dry Run →

ar[5] = 1 2 3 4 5

Size = 5	i = 0	i = 1
	i = 1	i = 2
	i = 2	i = 3
	i = 3	i = 4
	i = 4	i = 5

5.) find the Smallest Elemen of Array using User defined function.

Code :

```
#include<stdio.h>
int Smallest(int a[], int size);
int main(){
  int a[100], size, i;
  printf("Enter the size of Array : ");
  scanf("%d", &size);
  for(i=0; i<size; i++){
    printf("Enter the Element : ");
    scanf("%d", &a[i]);
  }
  Smallest(a, size);
  return 0;
}
int Smallest(int a[], int size){
  int s = a[0];
  for(i=0; i<size; i++){
    if(s>=a[i]){
      s=a[i];
    }
  }
  printf("Smallest Element of Array : %d", s);
}
```

FREEMIND

Date

Page 7

Output	Dry Run		
Enter the size of	a[5] = 5 4 3 2 1		
Array : 5	size = 5	i = 0	s = 5
Enter the element: 5		i = 1	s = 4
Enter the element: 4		i = 2	s = 3
Enter the element: 3		i = 3	s = 2
Enter the element: 2		i = 4	s = 1
Enter the element: 1			
Smallest Element of			
Array : 1			

FREEMIND

Date

Page 8

6.) Convert binary number to decimal number using user defined function.?

Code ⇒

```
#include<stdio.h>
int bintodec(int n);
int main(){
  int n;
  printf("Enter the Number :");
  scanf("%d",&n);
  bintodec(n);
return 0;
}
int bintodec(int n){
  int bin, rem, dec=0, base=1;
  bin=n;
while(n>0){
  rem=n%10;
  dec=dec+rem*base;
  n=n/10;
  base=base*2;
}
printf("The binary Number is %d", bin);
printf("The decimal Number is is %d", dec);
}
```

Dry Run ⇒

base=1	base=2	base=4	base=8	Output-
dec=0	dec=0	d=0	d=0	The binary Number is 1000
n=1000	n=100	n=10	n=1	
rem=1000%10=0	rem=0	rem=10%10 =0	rem=1%10=1	The decimal Number is is 8
d=0+0*1=0	dec=0+0*2=0	dec=0+0*4 =0	dec=0+1*8	
n=1000/10=100	n=100/10=0	n=10/10=1		
base=1*2=2	Base=2*2=4	base=4*2=8		

FREEMIND

Date

Page 9

7.) Demonstrate call by value and call by Reference

Call by Value :- In call by value, a copy of the actual parameter is passed to the functions. Any changes made to the parameters inside the function do not affect the original variables in the calling function.

Code :-

```
#include<stdio.h>
void Add(int a, int b);
int main(){
  int x, y;
  printf("Enter the two number :\n");
  Scanf("%d %d", &x, &y);
  int z = ~~xyz~~; Add(x,y);
    printf("Addition :", z);
}
void Add (int a, int b){
  int Sum;
  Sum= a+b;
  return sum;
}
```

Dry Run -	Add	Output :-
main		
x = 4	a = 4	Enter the two number:
y = 4	b = 4	4
z = 8	Sum = 4+4	4
	Sum = 8	Addition : 8

Call by Reference = In call by reference, the address of the actual parameters is paued to the function. Any changes made to the parameters inside the function affect the original variable in the calling function.

Code ⇒

```
#include <stdio.h>
void fnc(int *n);
int main(){
    int x = 10;
    printf("Before Calling Increment function, x = %d\n", x);
    fnc(&x);
    printf("After Calling increment function, x = %d", x);
    return 0; }
void fnc(int *n){
    (*n)++;
    printf("Inside increment function, *n = %d\n", *n);
}
```

Dry Run	Main	Inc	Output -
			Before Calling increment
x = 10		*n = 10	function, x = 10
		*n = 11	Inside Increment function, *n =
x = 11			11
			After calling Increment function,
			x = 11

FREEMIND

Date

Page 11

8.) Find the frequency of Character in a String.

Code ⇒

```
#include <stdio.h>
#include <string.h>
#define MAX 100
void fre(char str[MAX]);
int main(){
char s[MAX];
printf("Enter a string:");
gets(s);
fre(s);
return 0; }
void fre(char str[MAX]){
int i, len, freq[256]={0};
len = strlen(str);
for(i=0; i<len; i++){
if(str[i]!='\n')
freq[(char)str[i]]++; }
printf("Frequency of Characters in the string:\n");
for(i=0; i<256; i++){
if(freq[i]!=0){
printf("%c: %d times\n", (char)i, freq[i]);
}
}
}
```

FREEMIND

Date

Page 13

Assignment - 2

1.) To check whether a number given input by the user is prime or composite.

Code ⇒

```
#include<stdio.h>
int Prime (int n);
int main ()
{
  int num;
  printf("Enter a positive Integer: ");
  scanf("%d", &num);
  if (isPrime (num))
    printf("%d is a prime:\n", num);
  else
    printf("%d is Composite number:\n", num);
  return 0; }
int Prime (int n){
  int i;
  if (num<=1)
    return 0;
  for (i=2; i*i<=num; i++){
    if (num % i==0)
      return 0; }
  return 1; }
```

2.) To Print all the prime number in given range.

Code ⇒

```
#include<stdio.h>
int Prime (int n, int m);
int main ()
{
int num1, num2;
printf("Enter the two Range : ");
Scanf("%d %d", &num1, &num2);
Prime (num1, num2);
return 0; }
int Prime (int n, int m) {
    int f, i, j;
printf("Prime number from %d and %d are : \n", n, m);
for (i=num1+1; i<m; i++) {
    j = 0;
    for (j=2; j<=i/2; j++) {
        if (i%j==0) {
            f=1;
            break; }
        }
    if (f==0)
        printf("%d", i); }
}
```

3.) To check whether the given number is palindrome or not.

Code:

```
#include<stdio.h>
int Palindrome(int n);
int main(){
  int n;
  printf("enter the number:");
  scanf("%d",&n);
  if(Palindrome(n))
    printf("It is palindrome");
  else
    printf("It is not a palindrome:");
return 0;}
int Palindrome(int n){
  int t=n, d, r;
  r=0;
  while(n>0){
    d=n%10;
    r=(r*10)+d;
    n=n/10;}
if(r==t)
  return 1;
else
  return 0;}
```

4.) To find Sum of digit using User defined function.

Code →

```
#include <stdio.h>
int SD(int n);
int main(){
    int n=64;
    printf("%d", SD(n));
return 0; }
int SD(int n){
    int sum=0;
while (n!=0){
    sum=sum+n%10;
    n=n/10; }
retur sum; }
```

5.) To convert decimal to binary number using user defined function.

Code :

```
#include <stdio.h>
int DB (int n);
int main ()
{
int dec, bin;
printf ("Enter the decimal Number: ");
scanf ("%d", &dec);
bin= DB (dec);
printf (" Decimal (%d) = Binary (%d)", dec,
        bin);
return 0; }
in DB (int n)
{
int bin=0, rem, m, base=1;
m=n;
while (m != 0)
{
rem=m%2;
bin= bin+ base * rem;
base = base *10;
m= m/2;
return bin;
}
```

6.) To convert a decimal number to ~~hexadecimal~~ Octal and display the result:

Code ⇒

```
#include <stdio.h>
int DO(int n);
int main()
{
  int n;
  printf("Enter the number : ");
  scanf("%d", &n);
  DO(n);
  return 0; }
int DO(int n){
  int r, O=0, base=1;
  while (n>0){
    r= n%8;
    O = O+r*base;
    n = n/8;
    base = base*10; }
  printf("Octal Number : %d\n", O);
  return 0; }
```

Output ?

7.) To convert a decimal Number to hexadecimal and display the result.

Code :

```
#include <stdio.h>
int DH(int n);
int main(){
    int n;
    printf("Enter the number:");
    scanf("%d",&n);
    DH(n);
    return 0;}
int DH(int n);
    char H[20];
    int r, i=0;
    while (n!=0){
        r= n%10;
        if (r<10){
            H[i] = r+'0'; }
        else{
            H[i]= ~~remainder~~ r +55; }
        i++;
        n /=10; }
    printf("Hexadecimal Number:");
    for(int j=i-1 ;j>=0;j--){
        printf("%c", H[j]); }
    return 0; }
```

8.) To print table of a given Number

```
#include<stdio.h>
void Table(int n);
int main(){
   int n;
   printf("Enter the number:");
    scanf("%d", &n);
   printf("Table of %d is:\n", n);
   Table(n);
return 0; }
void Table (int n){
    int i;
for (i=1; i<=10; i++){
    printf("%d\n", (n*i)); }
}
```

9.) To find the factorial of a given number.

Code =>

```
#include<stdio.h>
int fact(int n);
int main(){
  int n, result;
  printf("Enter the number:");
  scanf("%d", &n);
  result = fact(n);
printf("%d! = %d", n, result);
  return 0; }
int fact(int n){
  int i, f=1
  for(i=1; i<=n; i++){
    f=f*i; }
    ret return f;
  }
```

10.) To find the length of a string.

Code :-

```
#include <stdio.h>
int str (char st[30]);
int main() {
  char st[30];
  int i, len;
  printf("Enter the string : \n");
  gets(st);
  len = str(st);
  printf("Length of given string is : %d", len);
  return 0; }
int str (char st[30]) {
    int i, len=0;
  for (i=0; st[i] != '\0'; i++) {
      len++; }
  return (len); }
```

2.) To Reverse the String using Stack.

Code :

```
#include <stdio.h>
#define MAX 20
int top = -1;
char stack[MAX];
char pop();
void push(char);
void main(){
  int i;
  char str[MAX], str1[MAX];
  printf("Enter the string :");
  gets(str);
  for(i=0;i<strlen(str);i++)
    push(str[i]);
  for(i=0;i<strlen(str);i++)
    str1[i]=pop();
  printf("Original string :");
  puts(str);
  printf("Revered string :");
  puts(str1);}
void push(char item){
  if(top==(MAX-1)){
    printf("Stack Overflow");
    return;}
  stack[++top]=item;}
char pop(){
  if(top==-1){
    printf("Stack Underflow");
    return 0;}
  return stack[top--];}
```

FREEMIND

Date

Page 24

Assignment - 3

1.)

Code ⇒

```
#include<stdio.h>
#define n 100
int stk[100], ch, n, top, x, i;
void push();
void pop();
void display();
int main(){
    top = -1;
    printf("Enter the size of stack : ");
    scanf("%d", &n);
    printf("\n 1. PUSH \n 2. POP \n 3. DISPLAY \n
           4. EXIT");
    while(1){
    printf("Enter the choice : ");
    scanf("%d", &choice);
    switch(ch){
    Case 1:
        push();
        break;
    Case 2:
        pop();
        break;
    Case 3:
        display();
        break;
    Case 4:
        printf("Exit");
```

```
break;
default:
   printf("Enter a Valid choice"); }
} return 0;
}

   void push(){
      if(top >= n-1)
         printf("Stack is Overflow");
      else {
       printf("Enter a value:");
       scanf("%d", &x);
        top++;
       stk[top] = x; }
          }

      void pop(){
        if (top <= -1)
           printf("Stack is Underflow");
        else {
         printf("The popped element in stack ~~is~~");
         ~~for~~ is %d", stk[top]);
         top--; }
        }
      void display(){
        if (top >= 0){
         printf("The Element in stack \n");
          for(i=top; i>=0; i--)
             printf("\n %d", stk[i]);
         printf("\n Press Next choice"); }
       else {
       printf("The stack is Empty"); }
             }
```

FREEMIND

Date

Page 27

3.) Check whether a given string is palindrome or not using Stack.

code ⇒

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define m 50
int top=-1, f=0;
int stk[m];
void push (char);
void pop ();
void main(){
int i, ch;
char s[m], b;
while (1){
printf("Enter the string Press 1 \n 2-exit");
printf("Enter your choice \n");
Scanf("%d", &ch);
Switch (ch){
Case 1:
printf("Enter string\n");
Scanf("%s", s);
for (i=0; s[i] != '\0'; i++){
b = s[i];
push (b); }
for (i=0; i<(strlen(s) /2); i++ ){
if (stk[top] == stk[i] ){
pop();
f++; }
else{
printf("It is not a palindrome\n");
break; } }
```

```
if ((strlen(s)/2)==i)
    printf("%s is palindrome",s);
    i=0
    top=-1;
    break;
case 2:
    exit(0);
default:
    printf("Invalid choice"); }
    } } }
void push (char a){
    top++;
    stk[top]=a; }
void pop (){
    top--; }
```

4.) Convert an infix expression to postfix using stack.

Code =>

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int prec(char c){
    if(c=='^')
        return 3;
    else if(c=='/'||c=='*')
        return 2;
    else if(c=='+'||c=='-')
        return 1;
    else
        return -1;}
char Assc(char c){
    if(c=='^')
        return 'R';
    return 'L';}
void ItoP(char s[]){
    char r[1000];
    int ri=0;
    int len=strlen(s);
    char stk[1000];
    int stkI=-1;
for(int i=0;i<len;i++){
        char c=s[i];
    if((c>='a'&&c<='z')||(c>='A'&&c<='Z')||
    (c>='0'&&c<='9')){
        r[ri++]=c;}
else if(c=='('){
    stk[stki++]=c;}
```

```
else if (c==')'){
    while (stk[i]>=0 && stk[stki]!='(')
    r[ri++]=stk[stki--]; }
    stki--; }
  else{
    while (stki>=0 && (prec(s[i])<prec(stk[stki])||
        prec(s[i]))==prec(stk[stki]) && Asso(s[i])=
        =='L'))
    r[ri++]=stk[stki--]; }
  stk[++stki]=c; }
    }
while (stki>=0){
    ~~result[r]~~ ~~r[ri]++~~
    r[ri++]=stk[stki--]; }
  r[ri]='\0';
  printf("%s\n",r); }
int main(){
  char exp[]="a+b*(c^d-e)^(f+g*h)-9";
    ItoP(exp);
  return 0; }
```

5.) Evaluate a given postfix expression using stack.

Code :

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#define m 100
int isempty(int top){
    return top==-1;}
int isfull(int top, int c){
    return top==c-1;}
void push(int stk[], int *top, int v){
  if(isfull(*top,m)){
    printf("Stack Overflow");}
    exit(Exit);}
  (*top)++
  stk[*top]=v;}
int pop(int stk[],int *top){
  if(isempty(*top)){
    printf("Stack underflow");
    exit(exit);}
  int v=stk[*top];
  (*top)--;
  return value;}
int EP(char *exp){
    int stk[m];
    int top=-1;
  for(int i=0; exp[i]!='\0';i++){
    if(isdigit(exp[i])){
    push(stk,&top,exp[i]-'0');}
else{
int op2=pop(stk,&top);
```

```
int op1 = pop(stk, &top);
switch(exp[i]){
  case '+':
  push(stk, &top, op1 + op2);
  break;
  case '-':
    push(stack, &top, op1 - op2);
  break;
  case '*':
    push(stk, &top, op1 * op2);
  break;
  case '/':
    push(stk, &top, op1 / op2);
  break;
  default:
    printf("Invalid Operation"); }
        } } } return pop(stk, &top);
    }
int main(){
  char exp[] = "35 2*+";
  printf("Postfix expression : %s\n", exp);
  printf("Result : %d\n", EP(exp));
  return 0; }
```

6.) Implement a Queue using array and show the implementation of all basic operation performed on queue.

Code ⇒

```
#include <stdio.h>
#include <stdbool.h>
#define m 100
int q[m];
int f=-1, r=-1;
bool isEmpty(){
    return (f==-1 && r==-1); }
bool isFull(){
    return (r+1)%m==f; }
void enq(int v){
    if(isFull()){
        printf("Queue is full");
        return; }
    else if(isEmpty()){
        f=r=0; }
    else{
        r=(r+1)%m; }
    q[r]=v;
int deq(){
    int v;
    if(isEmpty()){
        printf("Queue is Empty");
        return -1; }
    else if(f==r){
        v=q[f];
        f=r=-1; }
    else{
        v=q[f];
```

```
f = (f+1) % n; }
    return v; }
int peek() {
  if (isEmpty()) {
    printf("Queue is empty");
      return -1; }
    return q[f]; }
int main() {
     enq(10);
     enq(20);
     enq(30);
  printf("Dequeued element: %d\n", deq());
  printf("front element: %d\n", peek());
  printf("Is queue empty? %s\n", isEmpty()? "Yes": "No");
  return 0;
}
```

7.) Implement Queue Using Stack

Code →

```
#include <stdio.h>
#include <stdlib.h>
#define N 20
int s[N], top = -1;
int pop() {
  return s[top--]; }
void push (int x) {
  if (top == N-1)
    printf("Stack is Full");
  else {
    top++;
    s[top] = x; } }
void deqeng (int x) {
  push (x); }
void display() {
  int i;
  for (i=0; i<= top; i++)
    printf("%d", s[i]); }
int deq() {
  int data, res;
  if (top == -1)
    print ("Queue is Empty");
  else if (top == 0)
    return pop();
  data = pop();
  res = deq();
  push (data);
  return res; }
```

```
int main () {
  enq (5);
  enq (10);
  enq (15);
  enq (20);
  enq (25);
  printf("Queue:");
  display ();
  printf("\n Queue After Dequeue:");
  deq ();
  display (); }
```

FREEMIND

Date

Page 37

Assignment-4

1.) To implement Singly linked list and show all the operations performed on it.

Code ⇒

```
#include<stdio.h>
#include<stdlib.h>
Struct node{
    int data;
    Struct node *next; };
  Struct node *head;
void begi();
void lasti();
void randoi();
void begd();
void lastd();
void randod();
void display();
void search();
void main(){
int ch=0;
while (1){
  printf("\n1. Insert from begining\n2. Insert at last
\n3. Insert from random\n4. Delete from begining\n5. Delete
at last\n6. Delete from random\n7. Search\n8. Show
\n9. Exit\n");
  printf(" Enter the choice:");
  Scanf("%d",&ch);
  Switch(ch){
    case 1:
    begi(); break;
    case 2:
```

```
lastI(); break;
Case 3:
randoi(); break;
Case 4:
begd(); break;
Case 5:
lastd(); break;
case 6:
randod(); break;
Case 7:
Search(); break;
Case 8:
display(); break;
Case 9:
  exit(0); break;
default:
  printf("Please enter a Valid choice.:); } } }
void begi(){
Struct node *ptr;
 int itm;
ptr = (Struct node *) malloc (size of (struct node));
if (ptr == NULL){
   printf("\n Overflow"); }
else {
   printf("Enter value:\n");
  Scanf("%d", &itm);
   ptr → data = itm;
   ptr → next = head;
    head = ptr;
  printf("Node Inserted"); } }
void lasti(){
Struct node *ptr, *temp;
```

```
int item;
ptr = (struct node *) malloc (sizeof (struct node));
if (ptr == NULL){
    printf("\n Overflow");}
else{
    printf(" Enter a Value:");
    Scanf(" %d", &item);
    ptr -> data = item;
    if (head == NULL){
        ptr -> next = NULL;
        head = ptr;
        printf(" Node Inserted");}
    else{
        temp = head;
        while (temp -> next != NULL){
            temp = temp -> next;}
        temp -> next = ptr;
        ptr -> next = NULL;
        printf(" Node Inserted");}}}
void randoi(){
    int i, loc, item;
struct node *ptr, *temp;
ptr = (struct node *) malloc (sizeof (struct node));
    if (ptr == NULL){
        printf("\n Overflow");}
    else{
        printf(" Enter the Element to be Inserted:");
        Scanf("%d", &item);
        ptr -> data = item;
    printf(" Enter the location after:");
    Scanf("%d", &loc);
    temp = head;
```

```
for (i=0; i<loc; i++) {
    temp = temp -> ternext;
if (temp == NULL) { printf("Can't Insert\n");
    return; } }
ptr -> next = temp -> next;
temp -> next = ptr;
printf("\n Node Inserted "); }}
void beqd(){
Struct node *ptr;
if (head == NULL) printf("\n ~~Overflow~~ Underflow");
else {
    ptr = head;
    head = ptr -> next; free(ptr);
printf("\n~~ode~~ Node Deleted "); } }
void lastd(){
Struct node *ptr, *temp;
if (head == NULL){
    printf("\n List is empty"); }
else if (head -> next == NULL) {
    head = NULL;
    free(head); printf("Deleted"); }
else {
    ptr = head; while (ptr -> next != NULL) {
        ~~ptr~~ temp = ptr;
        ptr = ptr -> next; }
    temp -> next = NULL; free(ptr);
printf("Node Deleted"); }}
void randod(){
Struct node *ptr, *temp;
int loc, i;
printf("Enter the location:");
scanf("%d", &loc);
```

```
ptr = head; for (i=0; i<loc; i++){ temp = ptr;
    ptr = ptr -> next;
if (ptr == NULL){ printf(" Can't delete"); return; } }
 temp -> next = ptr -> next;
  free(ptr);
 printf("\n Node Deleted"); %d", loc+1); }
void Search(){
Struct node *ptr;
int itm; i=0, f; ptr = head; if (ptr == NULL){
 printf("\n Empty list"); }
 else { printf(" Enter item which you want to search ");
   Scanf("%d", &itm); while (ptr != NULL){
  if (ptr -> data == item){
printf(" item found at %d", loc+1); flag=0; }
else { f=1; } i++;
 ptr = ptr -> next; } if (f==1){ printf("Item not found");
        } } }
void display(){
  Struct node *ptr;
   ptr = head;
   if (ptr == NULL){
     printf(" Nothing to print"); }
   else {
     printf(" Linked List Elements: ");
     while (ptr != NULL){
       printf(" %d", ptr -> data);
        ptr = ptr -> next; }
            }
            }
```

2.) To Implement Stack using linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
Struct node {
    int data;
    Struct node *next; };
    Struct node *head;
void begi();
void begd();
Void display();
void main(){
 int ch = 0;
 while (1){
    printf("\n1. Insert or PUSH \n2. POP \n3. DISPLAY
        \n4. Exit");
    printf(" Enter the choice : ");
    Scanf("%d", &ch);
    Switch (ch){
      Case 1: begi();
        break;
      Case 2: begd(); break;
      Case 3: display(); break;
      Case 4: exit(0); break;
      default: printf(" Invalid choice "); } } }
void begi(){
  Struct node * ptr;
    int itm;
  ptr = (Struct node *)malloc (sizeof (Struct node));
  if ( ptr == NULL)
      printf(" Stack OVERFLOW ");
  else { printf(" Enter Value ");
      Scanf("%d", &itm);
```

```
    ptr→ data = item;
    ptr→ next = head;
    head = ptr;
    printf("\n Item Pushed in stack"); }}
void begd(){
    Struct node *ptr;
    if (head = NULL)
        printf("\n Stack Underflow");
    else { ptr = head;
        head = ptr→next; free(ptr);
printf("\n Item Popped from stack"); }}
void display (){
    Struct node *ptr;
    ptr = head;
    if (ptr == NULL)
        printf("\n stack is Empty");
    else {
        printf("\n Element in stack are :");
    while (ptr != NULL){
        printf("%d", ptr→data);
        ptr = ptr→next; } } }
```

3.) To Implement queue using linked list.

Code →

```
#include<stdio.h>
#include<stdlib.h>
Struct node {
int data;
struct node *next; };
Struct node *head;
void beg?(); void lastd(); void display();
void main(){
int ch=0; while(1){
printf("\n1. ENQUEUE \n2. DEQUEUE \n3. DISPLAY \4.
    EXIT \n");
printf(" Enter the choice: ");
Scanf("%d", &ch);
Switch(ch){
 Case 1:
  beg?(); break;
 Case 2: lastd(); breaks;
  Case 3: display(); break;
  Case 4: exit(0); break;
  default: printf(" Invalid choice "); } } }
void beg?(){
 Struct node *ptr; int itm;
 ptr = (struct node*) malloc(sizeof(struct node));
  if( ptr==NULL) printf(" OVERFLOW QUEUE");
else{
  printf("Enter the Value"); Scanf("%d", &itm);
  ptr→data=itm; ptr→next=head;
  head = ptr; printf(" Item Inserted"); } }
void lastd(){
Struct node *ptr, *temp;
 if (head==NULL) printf(" Queue is Empty");
```

```
else if (head->next == NULL) { head = NULL;
                    free(head);
printf("Item Deleted"); }
else { ptr = head; while (ptr->next != NULL) {
    temp = ptr;
        ptr = ptr->next; }
    temp->next = NULL; free(ptr);
printf("Item Deleted"); }}
void display (){ Struct node *ptr; ptr = head;
        if (head ptr == NULL) printf("Queue is Empty");
else {
    printf("Elements in Queue:");
while(ptr != NULL) {
printf("%d", ptr->data);
ptr = ptr->next; }}}
```

4.) To ~~ad~~ show the ~~St~~ addition of a polynomial using linked list.

Code ⇒

```
#include<stdio.h>
#include<stdlib.h>
    Struct node { int coef; int exp; Struct node *next; };
void insert(Node** poly, int coef, int exp){
Nod *temp = (Node*) malloc(sizeof(Node));
temp->coef = coef;
Temp->exp = exp; temp->next = NULL;
if (*poly == NULL) { *poly = temp; return; }
Node* current = *poly; while (current->next != NULL){
    current = current->next; }
current->next = temp; }
void print(Node* poly) { if (poly == NULL) {
    return 0; }
```

```
Node * current = poly; while (current != NULL) {
printf("%d.dx^%d", current → coef, current → exp);
if (current → next != NULL) { printf("+"); } current = current → next; }
printf("\n"); } Node * add (Node * poly1, Node * poly2) {
Node * result = NULL; while (poly1 != NULL && poly2 != NULL) {
if (poly1 → exp == poly2 → exp) {
insert (&result, poly1 → coef + poly2 → coef, poly1 → exp);
poly1 = poly1 → next; poly2 = poly2 → next; }
else if (poly1 → exp > poly2 → exp) {
insert (&result, poly1 → coef, poly1 → exp); poly1 = poly1 → next;
} else { insert (&result, poly2 → coef, poly2 → exp);
poly2 = poly2 → next; } }
while (poly1 != NULL) {
insert (&result, poly1 → coef, poly1 → exp); poly1 = poly1 → next;
} while (poly2 != NULL) { insert (&result, poly2 → coef, poly2 → exp);
poly2 = poly2 → next; } return result; }
int main() { Node * poly1 = NULL;
insert (&poly1, 5, 4);
insert (&poly1, 3, 2);
Node * poly2 = NULL;
insert (&poly2, 4, 4);
insert (&poly2, 2, 2);
printf("First Poly :"); printf(poly1);
printf("Second Poly:"); printf(poly2);
Node * result = add (poly1, poly2);
printf("Result:");
printf(result);
return 0;
}
```

5.) To create a doubly linked list with five nodes and display its elements.

code :

```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int n;
  struct node *pre;
  struct node *next; };
struct node *stnode, *ennode;
void DLC (int n);
void Display ();
int main (){
  int n;
  stnode = NULL;
  ennode = NULL;
printf("\n Input the number of node : ");
  scanf("%d", &n);
  DLC (n);
  Display ();  return 0; }
void DLC (int n) {
  int i, num;
  struct node *fnode;
if (n >= 1) {
stnode = (struct node *) malloc (sizeof (struct node));
if (stnode != NULL) {
 printf(" Input data for node 1: ");
 scanf("%d", &num);
stnode -> num = num;
stnode -> pre = NULL;
stnode -> next = NULL;
 ennode = stnode;
for (i = 2; i <= n; i++) {
```

```
fnnode = (struct node *) malloc (sizeof(struct node));
if (fnnode != NULL) {
printf("Input data for node :%d", 9);
scanf("%d", &num);
fnnode → num = num;
fnnode → pre = ennode;
fnnode → next = NULL;
 ennode → next = fnnode;
  ennode = fnnode; }
else { printf(" Memory not allocated");
      break; } } else {
      printf("Memory can't be allocated"); } } }
void Display ( ){
struct node *temp; int n = 1;
if (stnode == NULL){
   printf(" No data found in the list yet:"); }
 else { tmp = stnode;
printf("\n Data entered on the list are:\n");
   while (tmp != NULL){
   printf("node %d : %d\n", n, temp → num);
    n++;
tmp = temp → next; }
            }
        }
```

6) To create a circular linked list with five nodes and display its elements.

Code ⇒

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node * next; } Node;
Node * crN (int data) {
Node * newN = (Node *) malloc (sizeof (Node));
if (!newN) {
printf ("Memory Allocation failed \n"); exit(1);
}
newN -> data = data;
newN -> next = NULL;
return newN; }
void Display (Node * head) {
if (head == NULL) {
printf ("List is empty \n");
return; }
Node * temp = head;
do { printf ("%d", temp -> data);
temp = temp -> next; }
while (temp != head);
printf ("\n"); }
int main () {
Node * head = Creat crN (1);
Node * Second = crN(2);
Node * third = crN(3);
Node * fourth = crN(4);
Node * fifth = crN(5);
head -> next = Second;
Second -> next = third;
```

```
third next = fourth;
fourth -> next = fifth;
fifth -> next = head;
display (head);

Node * temp = head;
Node * nextNode;
do {
    nextNode = temp -> next;
    free (temp);
    temp = nextNode; }
while (temp != head);
    return 0; }
```

7.) To Implement linear Search on an Array.

Code ⇒

```
#include<stdio.h>
int LS(int *ar, int s, int k){
for (int i=0; i<size; i++){
    if (ar[i] == key) {
     return i; }
     }
return -1;
 }
int main (){
 int ar[10] = {3,4,1,7,5,8,11,42,3,13};
 int s = sizeof(ar)/sizeof(ar[0]);
 int k=4;
 int index = LS(ar,s,k);
if( index == -1){
   printf("The element is not present in the
      array "); }
else{
  printf("The element is present at arr :%d",
      index); }
return 0;
 }
```

8.) To Implement Binary search on an Array.

Code →

```
#Include<Stdio.h>
Pint BinS (int ar[], Pint l, Pint r, Pint x) {
 if (r>=1){
  int mid = l+ (r-1)/2;
 if (ar[mid] ==x)
      return mid;
 if (ar[mid] >x){
    return BinS (arr, l, mid -1, x);
   }
return ~~bin~~ BinS (arr, mid+1, r, x);
 }
 return -1;
  }
Pint main (){
  Pint ar[] = {2, 3, 4, 10, 40};
  int S[] = Size of (ar) / size of (ar[0]);
  Pint x = 10;
  Pint Pinden = BinS (~~arr, 0, S~~ ar, 0, S -1, x);
if (index ==-1) {
  printf(" Element is not present in array "); }
else {
  printf(" Element is present at inden ./.d", index);
}
return 0;
  }
```

Output?

31/05/24

Sikram
21.5.24