Date : 4/9/24 | Exp. Title | Page No. 1
Exp. No. 01

1. Write a menu driven to create a dynamic array of n elements and perform the following operations
a. Insert a new at a specified position
b. Delete an element at a specified position
c. Display.
d. Exit

```
#include <stdio.h>
int main()
{
  int *p, ele, ch, n, i, pos// p is pointer to create a dynamic
          array
  printf("Enter no. of elements to create an array:\t");
  scanf("%d", &n);
  p= malloc(n* sizeof(int)); // use malloc to create dynamic
                                                        array.
  printf("Dynamic array created\n");
  printf("Enter %d elements\n", n);
  for(i=0; i<n; i++) // read n the elements to array
  {
    scanf("%d", &p[i]);
  }
  while (1)  // repeats the menu
  {
  printf("\n 1.Insert \n2. Delete \n. Display \n 4.Exit\n
          Enter your choice: \t");
  scanf("%d", &ch);
  switch(ch).
  {
  case1: printf("\n Enter element and pos(0to%d)
              to insert:\t", n-1);
```

Date :	Exp. Title	Page No.
Exp. No.		2

```
Scanf ("%d%d", &ele, & pos);
realloc (P, (n+1)* size of (int)); // increase the size of array by 1
n= n+1 ; //update new size.
for (i=n-1; i>= pos; i--)// Start moving all the elements
{                                  to next position.
 P[i] = P[i-1] ;
}
P[POS] = ele ; //insert the new element at specified
break;                                              -position.
case 2: printf ("Enter position (0 to %d) to delete :\t", n-1);
   Scanf ("%d", &pos);
   for (i= pos+1 ; i<n ; i++)// delete the position element
   {          by moving next element of pos to next
    P[i-1] = P[i]          previous pos.
   }
   n=n-1; //update the count total element.
   brake ;
case 3:printf ("\n Array elements are : \n");
   for (i=0; i<n ; i++)
   {
    printf("%d\t", P[i]);
   }
   break;
case 4: exit (0);
 }
}
return 0;
}
```

05+05+20=25 (30)

Date: 9/10/24

Exp. No. 02

Exp. Title

2. Write a Menu driven program for the following operations
a. Create a sparse matrix.
b. Tranpose of Sparse matrix.
c. Exit

```
*/
#include <stdio.h>.
#include <stdlib.h>
#define MAX100.
Struct term
{
   int rows;
   int columns;
   int value;
};
 Struct term a[MAX], b[MAX]
 void create(); //declaring of a function
 void transpose();
  void display (int n, Struct term m[]);
 int main()
 {
  int Choice;
  while(1) // Continuosly repeats the menu
  {
    printf("\n Menu: \n");
   printf("1. Create Sparse matrix:\n");
   printf ("2. Transpose of Sparse matrix \n");
    print("3. Exit\n");
   printf ("Enter your Choice:");
```

Date :

Exp. No.

Exp. Title


```
scanf ("%d", &choice);
Switch (Choice)
{
 Case 1: Create ();
   break;
 Case 2: Transpose();
   break;
 Case 3: exit (0);
   }
 }
 return 0;
}
void create.
{
  //int matrix [10][10];
  int i, rows, coloums, n; //Starting index from 1 since
           0 is used for matrix dimensions.
  printf("\n Enter no. of rows, columns and no. of
           values: ");
  Scanf ("%d %d %d", &rows, &columns, &n);
  a[0]. rows = rows;
  a[0]. columns = columns;
  a[0]. value = n;
  for (i= 1; i<n ; i++)
  {
  printf ("\n Enter row, col and value: ");
  Scanf ( "%d %d %d", &a[i]rows , & a[i] .colums
           & a[i]. value);
  }
  display (n, a);
  }
```

```
void transpose ()
{
 int i,j,K=1,n;
  n= a[0].value;
 b[0].rows = a[0].columns;
 b[0].columns = a[0].rows;
 b[0].value = n;
 for (i=0; i<=a[0].columns; i++)
 {
   for (j=1; j<=n; j++)
   {
    if (a[j].columns ==i)
    {
     b[k].rows = a[j].columns;
     b[K].columns = a[j].rows;
     b[K] value = a[j].values;
    K++;
   }
  }
 }
  display (n,b);
}
void display (int values, struct term m[])
{
  int i;
 printf ("\n Row\t column\t value\n");
 for (i=0; i<=values; i++)
 {
printf ("%d \t %d\t %d\n", m[i].rows, m[i].columns,
                m[i].value);
  }
}
```

27-02-2020 28/30

3. Write a menu driven c program for the following Stack operations using arrays.
1. Push
2. POP.
3. Status
4. Display
5. Exit
Support all the operations with user defined function.

```
#include <stdio.h>
# include < stdlib.h>.
# define [MAX_SIZE 5
int Stack [MAX-SIZE];
int ele, top = -1; // global declaration
void push (int );        // declaring function.
int pop();
void Status ();
void display();
int main()          //driven program
{
  int Ch;
  while (1)       // repeats the menu.
  {
  printf ("\n menu 1. Push\n 2. POP \n 3. Status \n
              4. Display \n 5. Exit \n Enter your choice:");
  Scanf("%d", &ch);
  Switch (ch) // It is used for optional choice.
  {.
```

Date :	Exp. Title	Page No.
Exp. No.		7

```
case 1: printf ("Enter push elements:");
        scanf ("%d", &ele);
        push(ele); // push elements to function of push
        break; // breaks the case.
case 2: ele = pop ();
        printf ("popped %d", ele );
        break;
case 3: Status ( );
        break ();
case 4: display ();
        break
case 5: exit (0);
    }
  }
  return 0;
}
void push (int ele)   //local variable
 {
    if (top== MAX_SIZE -1)  // to check stack is empty
                              or not.
     {
      printf ("stack is full");
     }
    Else{
     Stack[++top]=ele;
     }
 }
int pop ( )
 {
```

Date :	Exp. Title	Page No. 8
Exp. No.		

```
if (top == -1)
{
 printf("stock is empty");
}
else
 {
  return stock (top--];
 }
}
void status ()
{
 if top == -1 )
 {
  printf ("stack is empty");
 }
 else if (top == MAX-SIZE -1)
  {
   printf (" stack is full");
  }
else {
   display ();
   }
}
void display ()
{
 printf (" Stack elements are : \n");
 for (i= top ; i>=0 ; i--)
 {
 printf ("%d \n" , stock[i]);
 }
}
```

27/30

$07+20$

Date : 23-10-24.

Exp. No. 4.

Exp. Title

4. Write a C program to Evaluate postfix/suffix Expression which contain only digit using Stack Support user defined function to Evaluate Postfix Expression

```
#include <stdio.h>
# include <math.h>  // for pow () function
# include<ctype.h>  // for is digit () function
# define max 10
int Stack [max];
int Top = -1 , Ele ;
void push (int);
int pop ();
void priority ( int, char , int );
int main (){
    int i , OP1 , OP 2;
    char Exp [20], symbal;
    printf ("Enter input valid sufix expression");
    Scanf( "%s", expression);
    for (i=0; expression[i]! = '\0'; i++)
    {
      Symbal = expression[i];
      if (isdigit (symbal))
      {
        push (symbal - '0');
      }
      else
      {
        op2 = pop();
        op1 = pop();
        evaluate (op1, symbal.op2);
      }
    }
```

Date :	Exp. Title	Page No.
Exp. No.		10

```
printf ("result=%d", pop());
return 0;
}
void push (int ele)
{
  Stack[++top]=ele;
}
int pop()
{
  return Stack[top--];
}
void evaluate (int OP1, char symbol, int OP2)
{
  int res;
  switch(symbol)
  {
    case'+':res = OP1 + OP2;
        push(res); break;
    case'-':res = OP1 - OP2;
        push(res); break;
    case'*':res = OP1 * OP2;
        push(res); break;
    case'/':res = OP1/OP2;
        push(res); break;
    case'%':res = OP1 % OP2;
        push(res); break;
    case'^':res = OP1^ OP2;
        push(res); break;
  }
}
```

Date : 30-10-24	Exp. Title	Page No. 11
Exp. No. 5		

5. Write a C program to convert given infix Expression to postfix Expression which contain alphanumeric operands & operators (+, -, *, /, %, ^,)

```
#include <stdio.h>
#include <ctype.h>
#define MAX_SIZE 10
char stack [20];
int top = 1;
void push (char ele);
char pop();
int priority (char sym);
int main ()
{
   int i = 0;
   char exp [20];
   char sym, ele;
   printf ("enter a valid infix expression:");
   scanf ("%s", exp );
   printf ("\n postfix:");
   for (i=0; exp[i] != '\0'; i++)
   {
      sym = exp[i];
      if (isalnum (sym))
      {
         printf ("%c", sym);
      }
      else if (sym == '(')
      push (sym);
      else if (sym == ')')
      {
```

Date :	Exp. Title	Page No.
Exp. No.		12

```
while ((ele=pop()!='('))
 printf("%c", ele);
}
else
{
 while (priority(Stack[top]) >= priority(sym))
  printf("%c", pop());
  push(sym);
 }
}
while (top != -1)
{
 print("%c", POP());
}
return 0;
}
void push(char ele)
{
 Stack[++top]=ele;
}
char pop()
{
  return Stack[top--];
}
 int priority(char sym)
 {
  if (sym == '(')
  return 0;
  if (sym == '+' || sym == '-'
  return 1
 if (sym == '*' || sym == '/' || sym == '%'
```

te :

Exp. Title

o. No.

```
return 2;
if ('sym == '^')
    return 3;
return 0;
}
```

[illegible]

Date :	Exp. Title	Page No.
Exp. No.		

6. write a menu driven c program to implement following circular Queue operations using Arrays.

1. Insert
2. Delete
3. Display
4. Exit

```
#include <stdio.h>
# include <stdlib.h>
# define MAXSIZE 5
int i, ele; rear =-1, front =-1, count=0;
int cqueue [MAXSIZE];
void insert (int ele);
int delete ();
 int main ()
 {
  int ch;
  while (1).
 {
 printf ("\n menu :\n 1.insert \n 2. delete \n 3. display
   \n4. Exit\n  Enter your choice :");
 scanf ("%d", &ch);
 switch (ch).
 {
case 1: printf ("enter the elements :\n");
        scanf ("%d ,&ele);
```

Exp. No.

```
insert (ele);
 break;
case2 : ele = delete ();
      printf ("queue elements deleted :%d ",ele);
      break;
case3 : display ();
      break;
case4: exit (0);
  }
 }
 return 0;
}
void insert (int ele)
{                         if ( count == MAXSIZE)
 printf ("Queue is full :\n");
 }
else
{
 rear = (rear+1) % MAXSIZE ;
 cqueue [rear] = ele;
 count ++;
 }
}
int delete ()
{
if (count == 0)
 {
 printf ("Queue is Empty !\n");
}
else
```

Date :

Exp. No.

Exp. Title

Page No.

```
{
  front = (front+1) % MAXSIZE;
  ele = (queue [front);
    count--;
   printf ("deleted element %d", ele);
  }
}
void display
{
 int i;
  printf ("Queue element are :\n");
   for ( i=front+1 ; i!= rear, i = (i+1) % MAXSIZE)
   {
    printf (%d ", (queue[i]);
   }
    printf ("%d ", (queue[i]);
  }
```

Date :	Exp. Title	Page No.
Exp. No. 7		

7.) Write a menu driven program for the following operations of single linked list student data with fields USN, Roll No, Name.

1.) Create SLL of n number of Students data by inserting front of the list.

2.) perform inserting / Deletion at end of SLL

3.) perform insertion / Deletion at front of SLL

4.) Display the status of SLL & count numbers of nodes init

5.) Exit

program:-

```
# include <stdio.h>
# struct Student
{
 int roll no;
  char USN [12];
  char name [50];
  Struct student * link;
};
 type def Struct Student * NODE;
  int C = D;
```

Date :
Exp. No.
Exp. Title
Page No.

```
NODE first = NULL;
NODE create node ();
void create SLL ();
void display SLL ();
void insert front ();
void insert end ();
void delete front ();
void delete end ();
    int main (){
    int ch;
     while (1) {
printf (" \n menu \n 1. Create SLL \n 2. Display SLL \n
 3. Insert front \n 4. Insert end \n 5. Delete front \n 6.
 Delete end \n 7. Exit \n.
Scanf (" %d", &ch);
 Switch (ch){
     case 1: Create SLL ();
              break;
     case 2: display SLL ();
              break;
     case 3: Insert front ( );
              break;
     case 4: Insert end ();
              break;
     case 5: delete front ();
              break;
     case 6: delete end ();
              breake.
     case 7: exit (0);
     }
}
```

Date :	Exp. Title	Page No.
Exp. No.		

```
  return0;
}
 NODE createnode () {
   NODE temp = malloc (size of (struct Student));
printf ("Enter roll number , usn, name : ");
scanf ("%d %s %s ", & temp->rollno, temp->USN, temp
      -> name);
 temp -> link = NULL;
  c++ ;
  return temp;
}
 void createsll (){
 int i, n;
 NODE temp;
 printf ("Enter number of Students :");
  Scanf ("%d ", &n);
  for (i=0; i <n; i++)
 {
  printf ("Enter student %d details : ", i+1);
   temp = create node ();
   if ( first = = NULL ) {
      first = temp;
    }
  }
 display SLL();
  }
 void display SLL () {
   NODE cur = first ;
   if ( first==NULL) {
 printf (" SLL is empty");
  return;
```

Date :	Exp. Title	Page No.
Exp. No.		

```
}
printf ("Student Details :\n");
while (cur != NULL){
printf ("%d\t %s \t %s \n", cur->rollno,cur->
usn, cur->name);
cur = cur -> link;
}
printf ("Number of Students is %d", c);
}
void insertfront (){
NODE temp = createnode ();
if (first ==NULL){
first = temp;
return;
}
temp -> link = first;
first = temp;
}
void insertend (){
NODE
cur = first, temp = create node ();
while ( cur -> link != NULL){
cur = cur -> link;
}
cur -> link = temp;
}
void deletefront (){
if (first == NULL){
printf ("SLL is Empty");
return;
}
```

Date :

Exp. Title

Page No.

Exp. No.

```
first = first -> link;
c--;
displayll();
}
void delete end )
NODE cur = first;
if (first == NULL)
{
printf ("SLL is Empty");
return;
}
if (first -> link == NULL)
{
first = NULL;
c--;
return;
}
while (cur -> link -> link != NULL)
{
cur = cur -> link;
}
cur -> link = NULL;
c--;
```

Date :

Exp. No.

Exp. Title

display LL ( );

}

Date :	Exp. Title	Page No.
Exp. No.		

9. Develop a menu driven program in C for the following operations on Binary Search tree (BST) of Integers.

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2.

b. Traverse the BST in Inorder, preorder & post order

c. Search the BST for a given element (Key) and report the appropriate message.

d. Exit

program.

```
# include <stdio.h>.
# include < stdlib.h>
struct node
{
struct node* left child;
int data;
struct node* right child;
};
typedef struct node * tree pointers;
tree pointer root = NULL; // Initalize an empty tree
tree pointer createNode (int value);
tree pointer insert BST (tree pointer root, int value);
    void inorder (tree pointer root);
    void preorder (tree pointer root);
    void postorder (tree pointer root);
```

Date :

Exp. No.

Exp. Title

Page No.

```
void search (tree pointer root, int key);
intmain ()
{
int values[] = {6,9,5,2,8,15,24,14,7,10};
int i, ch, key, n=10; //n is size of the above array
while (1)
{
printf ("1. create BST \n 2. Traversals \n 3. search\n 4.
    Exit \n Enter your choice :");
 scanf ("%d", &ch);
 switch (ch).
 {
 case:1:
     for (i=0; i<n; i++)
   {
   root = insert BST (root, values[i]);
}
printf ("Binary search tree constructed.\n");
 break;
case 2:
     printf ("\n Inorder :");
     inorder (root);
     printf ("\n preorder:");
     preorder (root);
     printf ("\n post order:");
     post order (root);
     printf ("\n");
     break;
```

Exp. No.

```
case 3:
  printf ("\n Enter the key to search:");
    scanf ( "%d", & key);
     search (root, key);
case4:
    exit (0);
  }
}
return 0;
}
 tree pointer createNode (int value)
 {
  tree pointer temp = malloc (size of (struct node));
 temp -> data = value;
 temp -> leftdata = NULL;
 temp-> right data = NULL;
     return temp;
}
tree pointer insert BST (tree pointer root , int value)
{
 if (root==NULL)
}
else if (value < root -> data)
{
root-> leftchild= insert BST (root -> left child, value);
 }
else
{
```

Date :
Exp. Title
Page No.
Exp. No.

```
root -> right Child = insert BST (root-> right child,
                                          value);
   }
return root;
 }
 void inorder (tree pointer root)
 {
 if (root != NULL)
{
inorder (root -> left Child); // visit left Child.
print f ("%d", root -> data); // visit root
inorder (root -> right child); // visit Right child
 }
}
 void preorder (tree pointer root)
 {
  if (root != NULL).
{
 printf ("%d", root -> data);
 preorder (root -> left child);
 preorder (root -> right(child);
  }
}
void post order (tree pointer root).
 {
 if (root != NULL).
 {
```

Date :

Exp. No.

Exp. Title

Page No.

```
postorder (root -> leftchild);
post order (root -> right child);
printf ("%d", root -> data);
}
}
void search (treepointer root, int key)
{
tree pointer temp;
temp = root;
While (temp != Null)
{
if (key == temp -> data) // compare the node data
                              with key.
{
printf ("Key found? \n");
return;
}
else if (key < temp->data);
{
temp = temp -> leftchild;
}
else
{
temp = temp -> right child;
}
}.
```

28

Date :	Exp. Title	Page No.
Exp. No. 10.		

Develop a program in C for the following operations on Graph (G) of cities ~~a~~ create a Graph of N cities using Adjacency matrix.
b. print all the nodes reactable from a given starting node in a digraph using DFS / BFS . method.

Program:-

```
#include <stdio.h>
# include <stdlib.h>.
#define MAX 10.
int graph [MAX][MAX];
int visited [MAX];
int Queue (MAX ];
int front = -1 , rear = -1;
// Create a graph using adjacency matrix values
 void Create Graph (int n)
{
 int i, j ;
printf (" Enter the adjacency matrix values:\n");
 for (i=0; i<n ; i ++)
{
 print (" Enter row %d : ", i+1);
  for (j=0 ;j<n ; j++)
{
 // Enter 0 if ~~des~~ edge not existed otherwise 1
  Scanf ("%d", & graph [i][j]
  }
 }
}
```

Date :
Exp. No.
Exp. Title
Page No.

```
// Recursive version of DFS
void DFS (int start, int n)
{
int i;
printf ("%d", start);
visited [start] = 1; // put 1 for starting vertex.
                        that visited.
for (i=0; i<n ; i++)
{
 if (!visited [i] && graph [start] [i] == 1)
{
 DFS (i,n); // call DFS recursively to reach next
                  vertex.
    }
  }
}
// BFS Implementation using queue
 void BFS (int start, int n).
{
 int i, vertex;
 printf ("%d", start);
visited [start] = 1; // put starting vertex visited.
queue [++rear] = start; // Insert into queue
                     visited first vertex.
while (front <= rear)
{
vertex = queue [front ++]; // pop the vertex visited.
```

Exp. No.

Page No.

```
for (i=0 ; i<n ; i++)
 {
if (!visited [i] && graph [vertex][i] ==1 ).
{
printf ("%d", i);
  visited [i] = 1;
 queue [++rear] = i;
   }
  }
}
}
int main ()
{
  int ch, i;
  int n, start;
  while (1)
{
printf ("\n 1. Create a Graph \n 2. DFS \n 3. BFS \n
      4. Exit \n Enter your choice: \n");
scanf ("%d", &ch);
switch (ch)
 {
 case 1:
   printf ("Enter the number of cities :");
   scanf ("%d", &n);
   create Graph (n);
   break;
```

Date :

Exp. No.

Exp. Title

Page No.

```
case 2 :
    printf("\n Enter the Starting node : ");
    scanf ("%d", & start);
    printf ("\n Nodes reachable from node %d using
                DFS :", start );

DFS  (start, n);
// Reset visited array for BFS.
 for (i=0 ; i<n ; i++)
 {
 visited [i] = 0;
 }
break;
case 3 :-
        front = rear = -1 ; // reset queue for BFS
     printf ("\n nodes reachable from node %d
                using BFS :", start );
        BFS  (start,n );
       // Reset visited array for BFS.
         for (i=0; i<n ; i++ )
           {
             visited [i] = 0;
           }
             printf ("\n");

case 4:- exit(0);
            break;
       }
   }
return 0;
}
```

Exp. No. 11

11. Give a file of N employee records with a set of K of Keys (4-digit) which uniquely determine the record in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory address (2-digit) of locations in HT. let the keys in K and addresses in L are Integers

* Develop a program in C that uses Hash function K->L as H(K) = K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing. */

```
#include <stdio.h>
#include <stdlib.h>
#define MAX-Employees 5
# define HT_SIZE10

 Struct Employee
 {
 int Key;
 char name [30];
 };
Struct Employee Hash table
{
Struct Employee * employee [MAX_EMPloyees];
};
int ~~Hash~~ (int Key, int m)
```

Exp. No.

```
{
return Key % m;
}
void insert {Struct EmployeeHashTable *ht, Stouct
              Employee * emp, int m)
{
int index = hash (temp -> Key, m);
while (ht -> employees [index ]! = Null)
{
  index = (index + 1) % m;
}
ht -> employees [index] = emp;
}
void display (Struct EmployeeHashTable *ht, int m)
{
  int i;
  printf ("Hash Table :\n");
  for (i=0; i< m; i++)
  {
   if (ht ->employees [i] != Null )
{
printf ("Index %d: Key =%d, Name = %s \n",
          i, ht -> employees [i] -> Key
             ht -> employees [i] -> Name);
}
else
{
```

Date :

Exp. No.

Exp. Title

Page No.

```
printf ("Index %d: Empty \n", i);
  }
 }
}
intmain ()
{
 int i, m;
 Struct EmployeeHashTable ht;
 for (i=0; i<max_Employees ;i++)
  {
   ht. employee [i] =Null;
  }
 m =H7_SIZE;
 Struct Employee e1 = {1000, "Noor"};
 Struct Employee e2 = {1001, "rehman"};
 Struct employee e3 = {1002, "Altu"};
 Struct employee e4 = {1003, "Shanawaz"};
  insert (&ht, &e1, m);
  insert (&ht, &e2, m);
  insert (&ht, &e3, m);
  insert (&ht, m);
 return0;
}
```

28