S.Y. B.C.A. (Science) (Semester-III) Practical Examination CA 202 MJP: Data Structure Slip Solution

For Private Circulation

Slip-01

Q.1 Write a C program to search an element by using binary search method.

```
// Binary search Program
#include <stdio.h>

// Function to perform binary search
int binarySearch(int arr[], int size, int target) {
    int Begin = 0, End = size - 1,mid;

    while (Begin <= End)
    {
         mid = (Begin + End) / 2;

         printf("%d",mid);

        // Check if target is at mid
        if (arr[mid] == target)
            return mid;

        // If target is greater, ignore left half
        else if (arr[mid] < target)
            Begin = mid + 1;

        // If target is smaller, ignore right half
        else
            End = mid - 1;
    }

    // Element not found
    return -1;
}

int main() {
    int arr[100], n, target;

    // Input size of array
    printf("Enter number of elements (sorted): ");
    scanf("%d", &n);

    // Input elements
    printf("Enter %d sorted elements:\n", n);
    for (int i = 0; i < n; i++)
```

```
        scanf("%d", &arr[i]);

    // Input target value
    printf("Enter element to search: ");
    scanf("%d", &target);

    // Perform binary search
    int result = binarySearch(arr, n, target);

    // Output result
    if (result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found in the array.\n");

    return 0;
}
```

Q.2) Write a C program to implement static stack of integer with operations:

● Push () ● Pop () ● Empty ()

```
// Implementation of static stack

 #include<stdio.h>
 #define MAX 5
 #define EMPTY -1
 #define FULL MAX-1

  typedef struct stk
  {
         int top;
         int item[MAX];
  }stack;

  // -------------------initstack------------------------------

  void initstack(stack *ps)
  {
         ps->top=-1;
  }
  //--------------------------------------------------------------
  int isempty(stack *ps)
  {
           return(ps->top==EMPTY);
  }
  //-------------------------------------------------------------



 int isfull(stack *ps)
  {
          return (ps->top==FULL);
  }
  //-------------------------------------------------

  void push( stack *ps,int n)
  {
         if(isfull(ps))
         {
          printf("\n stack full");
         }
         else
         {
         ++ps->top;
         ps->item[ps->top]=n;
         }
  }

 //-------------------------------------------------
  int pop(stack *ps)
  {   int n1;
          n1=ps->item[ps->top];
          ps->top-- ;
          return n1;
  }


 //---------------------------------------------
 void show(stack *ps)
 {

  int t;
  t=ps->top;
  while(t>=0)
  {
     printf("%d  ",ps->item[t]);
     t--;
  }
 }

 int main()
 {
    int ch,n,n1;
        stack s1;

         initstack(&s1);

         do
         {
           printf("\n------------MENU---------------------------");
            printf("\n 1) Push element");
            printf("\n 2) POP element");
            printf("\n 3) Show element");
            printf("\n 4) Exit");
            printf("\n Enter your choice ");
            scanf("%d",&ch);
            switch(ch)
            {
             case 1:
                printf("\n Enter the element to insert :  ");
                scanf("%d",&n);
```

```
               push(&s1,n);
               break;



             case 2:
               if(!isempty(&s1))
               {
               n1=pop(&s1);
               printf("\n %d is deleted from stack",n1);
               }
               else
               {
                   printf("\n Stack is empty \n");
               }
               break;

            case 3:
               printf("\n The element in stack are as follows \n");
               show(&s1);
               break;

               case 4:
               break;
           }
        }while(ch!=4);
        return 0;
}


/*   ------------------------  Output --------------------------


[root@localhost DS-2015-16]# gcc stack.c
[root@localhost DS-2015-16]# ./a.out

--------------------MENU---------------------------
 1) Push element
 2) POP element
 3) Show element
 4) Exit
 Enter your choice 1

 Enter the element to insert :  3

--------------------MENU---------------------------
 1) Push element
 2) POP element
 3) Show element
 4) Exit
 Enter your choice 1

 Enter the element to insert :  5

--------------------MENU---------------------------
 1) Push element
 2) POP element
 3) Show element
 4) Exit
 Enter your choice 3

 The element in stack are as follows
5  3
--------------------MENU---------------------------
 1) Push element
 2) POP element
 3) Show element
 4) Exit
 Enter your choice 2

 5 is deleted from stack
--------------------MENU---------------------------
 1) Push element
 2) POP element
 3) Show element
 4) Exit
 Enter your choice 4
[root@localhost DS-2015-16]#

*/
```

Slip-02

Q.1) Write a C program to sort n elements using Bubble Sort.

```
#include<stdio.h>
// function for bubble sort
Bubblesort(int x[],int n)
{
int t,i,j;
for(i=0;i<n;i++)
{
for(j=0;j<n-1;j++)
{
if(x[j] > x[j+1])
{

t = x[j];
x[j] = x[j+1];
x[j+1] = t;
} //if end
} //j end
} //i end
}
//------------------------------------------------------------
int main()
{
int x[20],n,i;
printf("\n\n--------------Bubble Sort--------------------\n\n");
printf("\n Enter how many number u want to enter :----> ");
scanf("%d",&n);
printf("\n\n Enter all array element\n\n");
for(i=0;i<n;++i)
{
scanf("%d",&x[i]);
}
Bubblesort(x,n);
printf("\n\n------------The Sorted array --------------\n\n");
for(i=0;i<n;++i)
{
printf("%d ",x[i]);
}
return;
```

Q.2) Write a C menu driven program to implement doubly linked list of integers with following Operations:
● Create ● Delete ● Insert ● Display

```
#include<stdio.h>

  typedef struct node
  {
     int data;
     struct node *next,*prev;
  }NODE;


//-----------create function()-----------------------

void createlist(NODE *head)
{
   int n ,i;
   NODE *last,*newnode;

   printf("\n Enter how many nodes :  ");
   scanf("%d",&n);

   last = head;

   for(i=1;i<=n;i++)
   {

       newnode  =  (NODE *) malloc (sizeof(NODE));
       newnode->next=newnode->prev=NULL;

       printf("\n enter the node data");
       scanf("%d",&newnode->data);

       last->next = newnode;
       newnode->prev=last;
       last=newnode;
  }

 }


//-------------display()--------------------


void display(NODE *head)
```

```
{
  NODE *temp;

  for(temp=head->next;temp!=NULL;temp=temp->next)
  {
     printf("%d\t",temp->data);
  }

}

//--------------------Insert function--------------------
void insert(NODE *head,int num,int pos)
{
    NODE *newnode,*temp,*temp1;
    int i;

   for(temp=head,i=1;(temp!=NULL)&&(i<=pos-1);i++)
       temp=temp->next;

   if(temp==NULL)
   {
     printf("\n Position is out of range");
     return;
  }

  newnode = (NODE *) malloc(sizeof(NODE));
  newnode->data = num;

 newnode->next  = newnode->prev =NULL;
 temp1=temp->next;
 newnode->next=temp1;
 temp1->prev=newnode;
 temp->next=newnode;
 newnode->prev=temp;

}

//-------------------------delete by position ------------------

void deletepos(NODE *head,int pos)
{
   NODE *temp,*temp1;

   int i;
   for(temp=head,i=1;(temp->next!=NULL)&&(i<=pos-1);i++)
         temp=temp->next;

   if(temp->next==NULL)
   {
        printf("\nPosition is out of range ");
        return;
    }

    temp1=temp->next;
    temp->next=temp1->next;
    if(temp1->next!=NULL)
     temp1->next->prev=temp;
    free(temp1);

}

//-------------------------delete by element--------------------
void deletevalue(NODE *head,int num)
{

NODE  *temp,*temp1;

for(temp=head;temp->next!=NULL;temp=temp->next)
  if(temp->next->data==num)
  {
     temp1=temp->next;
     temp->next = temp1->next;

    if(temp1->next!=NULL)
      temp1->next->prev=temp;

    free(temp1);

    return;
  }

printf("Element not found");

}

//--------------------main()------------------------


void main()
{
   NODE *head;
   int ch,n,pos;

   head=(NODE *) malloc (sizeof(NODE));

   do
   {
        printf("\n 1: CREATE ");
        printf("\n 2: INSERT");
        printf("\n 3: DISPLAY");
        printf("\n 4: DELBYPOS");
        printf("\n 5: DELBYVALUE");
        printf("\n 6: EXIT");

        printf("\nenter your choice  : ");
        scanf("%d",&ch);

        switch(ch)
        {

           case 1:
                  createlist(head);
                   break;
            case 2:
                 printf("\n Enter the element and position ");
                 scanf("%d%d",&n,&pos);
                 insert(head,n,pos);
                 display(head);
                 break;
          case 3 :
                display(head);
                 break;

          case 4:
                printf("\n Enter the position to delete");
                scanf("%d",&pos);
                deletepos(head,pos);
                display(head);
                break;
        case 5:
                printf("\n Enter the elementto delete");
                scanf("%d",&n);
                deletevalue(head,n);
                display(head);
                break;

        }
    }while(ch!=6);



}
```

Slip 03

Q.1) Write a C program to sort n numbers using insertion sort integers.

```
#include<stdio.h>
Insertsort(int x[],int n)
{
int next,i,newelement;
for(next=1;next<n;++next)
{
//newelement is element to be inserted
newelement=x[next];
// shift element > newelement to right by 1 pos
for(i=next-1;i>=0 && newelement<x[i];i--)
{
x[i+1]=x[i];
}
//insert new element at pos i+1
x[i+1]=newelement;
}
}
//----------------------------------------------------------

int main()
{
int x[20],n,i;
printf("\n\n--------------Insertion Sort--------------------\n\n");
printf("\n Enter how many number u want to enter :----> ");
scanf("%d",&n);
printf("\n\n Enter all array element\n\n");
for(i=0;i<n;++i)
{

scanf("%d",&x[i]);
}
Insertsort(x,n);
printf("\n\n--------------------The Sorted array --------------
\n\n");
for(i=0;i<n;++i)
{
printf("%d ",x[i]);
}
}
```

Q.2) Write a C program to accept an infix expression and convert it into postfix form.

```
#include<stdio.h>

#define MAX 5
#define EMPTY -1
#define FULL MAX-1

typedef struct stk
{
      int top;
      int item[MAX];
}stack;

// -------------------initstack-----------------------------

void initstack(stack *ps)
{
      ps->top=-1;
}
//---------------------------------------------------------------
int isempty(stack *ps)
{
        return(ps->top==EMPTY);
}
//--------------------------------------------------------------

int isfull(stack *ps)
{
       return (ps->top==FULL);
}
//----------------------------------------------------------------

void push( stack *ps,int n)
{
      if(isfull(ps))
      {
       printf("\n stack full");
      }
      else
      {
      ++ps->top;
      ps->item[ps->top]=n;
      }
}
```

```
 //------------------------------------------------------------

  int pop(stack *ps)
  {   int n1;
         n1=ps->item[ps->top];
         ps->top-- ;
         return n1;
  }

//--------Conversion of infix to postfix ---------------------

  void postfix(char in[],char post[])
  {
       int i,j=0;
       char ch;
       stack s1;
       initstack(&s1);
       for(i=0;in[i]!='\0';i++)
       {

          if(isalpha(in[i]))
           {
              post[j]=in[i];
              j++;
           }
           else
           {
             switch(in[i])
              {
        case '+' :
        case '-' :
        case '*' :
        case '/' :
        case '%' :
        case '(' :
              push(&s1,in[i]);
              break;
        case ')' :
              while((ch=pop(&s1))!='(')
              {
                     post[j]=ch;
                     j++;
              }
              }
            }
          }
         while(!isempty(&s1))
         {
              post[j]=pop(&s1);
              j++;
         }
         post[j]='\0';
       }

  //----------------------main-----------------------------

  void main()
  {

         char in[20],post[20];

         printf("\n Enter Infix string : ");
         scanf("%s",in);
         fflush(stdin);
         postfix(in,post); //--------------- postfix conversions
         printf(" \n Postfix sring is ");
         printf("%s",post);
        }
```

Slip 04

Q.1) Write a C program to search an element using linear search method.

```c
#include <stdio.h>

int linearSearch(int arr[], int n, int key) {

    for (int i = 0; i < n; i++) {

        if (arr[i] == key)

            return i;  // Return the index where key is found

    }

    return -1;  // Key not found

}

int main() {

    int arr[100], n, key, result;

    printf("Enter number of elements: ");

    scanf("%d", &n);

    printf("Enter %d elements:\n", n);

    for (int i = 0; i < n; i++)

        scanf("%d", &arr[i]);

    printf("Enter the element to search: ");

    scanf("%d", &key);

    result = linearSearch(arr, n, key);

    if (result == -1)

        printf("Element not found.\n");

    else

        printf("Element found at index %d (position %d).\n", result,
result + 1);

    return 0;

}
```

Q.2) Write a C menu driven program to implement singly circular linked list of integers with Following operations: ● Create ● Insert ● Delete ● Display

```
/* Implementaytion of Singly Circular Linked List */

 #include<stdio.h>

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


//------------create function()-----------------------

void createlist(NODE *head)
{
   int n ,i;
   NODE *last,*newnode;

   printf("\n Enter how many nodes :  ");
   scanf("%d",&n);

   last = head;

   for(i=1;i<=n;i++)
   {

       newnode  =  (NODE *) malloc (sizeof(NODE));
       newnode->next=head;

       printf("\n enter the node data  : ");

       scanf(" %d",&newnode->data);

       last->next = newnode;

       last=newnode;
  }

 }
```

```
//--------------display()---------------------


void display(NODE *head)
{
  NODE *temp;

  for(temp=head->next;temp!=head;temp=temp->next)
  {
     printf("%d\t",temp->data);
  }

}

//-----------INSERT function------------------------------------

void insert(NODE *head,int num,int pos)
{
   NODE *newnode,*temp;
  int i;

   for(temp=head,i=1;(temp->next!=head) &&(i<=pos-1);i++)
         temp=temp->next;

    if(temp==NULL)
    {
          printf("\n Postion is out of range ");
          return;
    }
    newnode = (NODE *) malloc (sizeof(NODE));
    newnode->data=num;
    newnode->next=temp->next;
    temp->next = newnode;
}

//---------------------delete by position ---------------------

void deletepos(NODE *head,int pos)
{
   NODE *temp,*temp1;

   int i;
   for(temp=head,i=1;(temp->next!=head)&&(i<=pos-1);i++)
         temp=temp->next;
   if(temp->next==NULL)
   {
       printf("\nPosition is out of range ");
       return;
   }

   temp1=temp->next;
   temp->next=temp1->next;

   free(temp1);

}


//--------------------main()------------------------


void main()
{
   NODE *head;
   int ch,n,pos;

   head=(NODE *) malloc (sizeof(NODE));

   do
   {
        printf("\n 1: CREATE ");
        printf("\n 2: DELETE BY POSITION");
        printf("\n 3: DISPLAY");
        printf("\n 4: INSERT");
        printf("\n 5: EXIT");

        printf("\nenter your choice  : ");
        scanf("%d",&ch);

        switch(ch)
        {

           case 1:
                  createlist(head);
                   break;
            case 2:
                 printf("\n Enter the position ");
                 scanf("%d",&pos);
                 deletepos(head,pos);
                 display(head);
                 break;
          case 3 :
                display(head);
                 break;




         case 4:
             printf("\n enter the element and position to insert");
               scanf("%d%d",&n,&pos);
                insert(head,n,pos);
               display(head);
               break;
        }

    }while(ch!=5);


}
```

Slip 05

Q.1) Write a C program to create and display singly linked list.

```
/* Implementaytion of Singly Circular Linked List */

 #include<stdio.h>

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


//------------create function()-----------------------

void createlist(NODE *head)
{
   int n ,i;
   NODE *last,*newnode;

   printf("\n Enter how many nodes :  ");
   scanf("%d",&n);

   last = head;

   for(i=1;i<=n;i++)
   {

       newnode  =  (NODE *) malloc (sizeof(NODE));
       newnode->next=NULL;

       printf("\n enter the node data  : ");

       scanf(" %d",&newnode->data);

       last->next = newnode;

       last=newnode;
  }

 }
```

```
//--------------display()---------------------


void display(NODE *head)
{
  NODE *temp;

  for(temp=head->next;temp!=NULL;temp=temp->next)
  {
     printf("%d\t",temp->data);
  }

}



//--------------------main()------------------------


void main()
{
   NODE *head;
   int ch,n,pos;

   head=(NODE *) malloc (sizeof(NODE));


                  createlist(head);
                  display(head);
}
```

Q.2) Write a C program to create BST and display its preorder , in-order traversal.

```
#include<stdio.h>
#define NODEALLOC (struct node *) malloc (sizeof (struct node ))
typedef struct node
{
int data;
struct Node *left,*right;
}NODE;
// ------function for creating BST------
NODE * createbst(NODE *root)
{
NODE *newnode,*temp;
int num,n,i;
printf("\n Enter how many node \n");
scanf("%d",&n);
for(i=1;i<=n;++i)
{
newnode=NODEALLOC;
printf(" \n Enter the data to insert ");
scanf("%d",&num);
newnode->data=num;
newnode->left=NULL;
newnode->right=NULL;
if(root==NULL)
root=newnode;
else
{
temp=root;
while(temp!=NULL)
{
if(num < temp->data)
{
if(temp->left==NULL) // not left child
{
temp->left=newnode;
break;
}
else
temp=temp->left; // move left
}
else
if(num > temp->data)
{
```

```
if(temp->right == NULL)
{
temp->right=newnode;
break;
}
else

temp=temp->right;
}
} // end while
}
} // for end
return root;
}

void preorder(NODE * root)
{
NODE *temp=root;
if(temp!=NULL)
{
printf("%d ",temp->data);
preorder(temp->left);
preorder(temp->right);
}
}
//-------------------------------------------------

void inorder(NODE * root)
{
NODE *temp=root;
if(temp!=NULL)
{
inorder(temp->left);
printf("%d ",temp->data);
inorder(temp->right);
}
}
//-------------------------------------------------------------
void postorder(NODE* root)
{
NODE* temp=root;
if(temp!=NULL)
{
postorder(temp->left);
postorder(temp->right);
printf("%d ",temp->data);
```

```
}
}
//-----------------------------main fun()---------------------
int main()
{
NODE *root=NULL;
int count;
root= createbst(root);

printf("\n Node in Preorder \n\n ");
preorder(root);
printf("\n Node in Inorder \n\n");
inorder(root);
printf("\n Node in Postorder \n\n ");
postorder(root);
return 0;
}
```

Slip 06

Q.1) Write a C program to reverse a string using Stack .

```
// Program for reverse a string using stack.
#include<stdio.h>
#define MAX 5
#define EMPTY -1
#define FULL MAX-1
typedef struct stk
{
int top;
int item[MAX];
}stack;
// -------------------initstack------------------------------
void initstack(stack *ps)
{
ps->top=-1;
}
//---------------------------------------------------------------
int isempty(stack *ps)
{
return(ps->top==EMPTY);
}
//--------------------------------------------------------------

int isfull(stack *ps)
{
return (ps->top==FULL);
}
//----------------------------------------------------------------
void push( stack *ps,int n)
{
if(isfull(ps))
{
printf("\n stack full");
}
else
{
++ps->top;
ps->item[ps->top]=n;
}
}
//-------------------------------------------------------------------
-i
int pop(stack *ps)
```

```
{ int n1;
n1=ps->item[ps->top];
ps->top-- ;
return n1;
}

//-----------------------------------------------------------------
----
void show(stack *ps)
{

int t;
t=ps->top;
while(t>=0)
{
printf("%d ",ps->item[t]);
t--;
}
}
int main()
{
stack s1;
char str[20];
int i=0;
initstack(&s1);
printf("\n enter the string\n");
gets(str);
//scanf("%s",str);
while(str[i]!='\0')
{
push(&s1,str[i]);
i++;
}
i=0;
while(!isempty(&s1))
{
str[i]=pop(&s1);
i++;
}
str[i]='\0';
printf("\n The reversed of string is :");
printf("%s\n",str);
return 0;
}
```

Q.2) Write a C program to read the data from the file "employee.txt" which contains empno and empname and sort the data on names alphabetically (use strcmp) using Bubble Sort.

```
#include<stdio.h>
#include<string.h>
typedef struct
{
char ename[30];
int eno;
}RECORD;
RECORD emp[100];
//-----------------------------------------------------------
int readFile(RECORD a[100])
{
int i=0;
FILE *fp;
if((fp=fopen("empinfo.txt","r"))!=NULL)
{
while(! feof(fp))
{
fscanf(fp,"%s%d", a[i].ename, &a[i].eno);
i++;
}
}
return i; // number of records read
}
//-----------------------------------------------------
void writeFile(RECORD a[100], int n)
{
int i=0;
FILE *fp;
if((fp=fopen("sortedemp.txt","w"))!=NULL)
{
for(i=0;i<n; i++)
{
fprintf(fp,"%s\t%d\n", a[i].ename, a[i].eno);
}
}

}
//-----------------------------------------------------
int Bubblesort(RECORD *a, int n)
{
int i,j;
RECORD t;
for(i=0;i<n;i++)
{
for(j=0;j<n-1;j++)
{
if(strcmp(a[j].ename , a[j+1].ename) > 0)
{
t = a[j];
a[j] = a[j+1];
a[j+1] = t;
} //if end
} //j end
} //i end
return 0;
}

//
int main()
{
int n;
n=readFile(emp);
Bubblesort(emp,n);
writeFile(emp,n);
return 0;
}
```

Slip 08
Q.1) Write a C program to create and display doubly linked list.

```
#include<stdio.h>

  typedef struct node
  {
     int data;
     struct node *next,*prev;
  }NODE;


//-----------create function()-----------------------

void createlist(NODE *head)
{
   int n ,i;
   NODE *last,*newnode;

   printf("\n Enter how many nodes :  ");
   scanf("%d",&n);

   last = head;

   for(i=1;i<=n;i++)
   {

       newnode  =  (NODE *) malloc (sizeof(NODE));
       newnode->next=newnode->prev=NULL;

       printf("\n enter the node data");
       scanf("%d",&newnode->data);

       last->next = newnode;
       newnode->prev=last;
       last=newnode;
  }

 }


//-------------display()--------------------


void display(NODE *head)
{
  NODE *temp;
  for(temp=head->next;temp!=NULL;temp=temp->next)
  {
     printf("%d\t",temp->data);
  }

}


//--------------------main()------------------------


void main()
{
   NODE *head;
   int ch,n,pos;

   head=(NODE *) malloc (sizeof(NODE));

                  createlist(head);
                 display(head);



}
```

Q.2) Write a C program to read the data from the file “person.txt” which contains person no and person age and sort the data on age in ascending order using insertion Sort.

```
#include<stdio.h>
#include<string.h>
typedef struct
{
int empid;
int age;
}RECORD;
RECORD emp[100];
//-----------------------------------------------------------
int readFile(RECORD a[100])
{
int i=0;
FILE *fp;
if((fp=fopen("emp.txt","r"))!=NULL)
{
while(! feof(fp))
{
fscanf(fp,"%d%d", &a[i].empid, &a[i].age);
i++;
}
}
return i; // number of records read
}
//----------------------------------------------------
void writeFile(RECORD a[100], int n)
{
int i=0;
FILE *fp;
if((fp=fopen("sortedemp.txt","w"))!=NULL)
{
for(i=0;i<n; i++)
{
fprintf(fp,"%d\t%d\n", a[i].empid, a[i].age);
}
}
}

//----------------------------------------------------
Insertionsort(RECORD a[100], int n)
{
int next,i;
RECORD newelement;
for(next=1;next<n;++next)
{
//newelement is element to be inserted
newelement=a[next];
// shift element > newelement to right by 1 pos
for(i=next-1;i>=0 && newelement.age < a[i].age;i--)
{
a[i+1]=a[i];
}
//insert new element at pos i+1
a[i+1]=newelement;
}
}

//-------------------------------------------------
int main()
{
int n;
n=readFile(emp);
Insertionsort(emp,n);
writeFile(emp,n);
return 0;
}
```

Slip 11

Q.1) Write a C program to generate n random numbers and sort it using QuickSort.

```
#include<stdio.h>

int x[20];
void main()
{
int n,i;
printf("\n\n--------------Quick Sort--------------------\n\n");
printf("\n Enter how many number u want to enter :----> ");
scanf("%d",&n);
printf("\n\n Enter all array element\n\n");
for(i=0;i<n;++i)
{
scanf("%d",&x[i]);
}
Qsort(0,n-1);
printf("\n\n--------------------The Sorted array -------------
\n\n");
for(i=0;i<n;++i)
{
printf("%d ",x[i]);
}
}
//-----------Function for Quick sort----------------------
Qsort(int m,int n)
{
int down,up,temp,pivot;
if(m<n)
{
down=m+1;
up=n;
pivot=x[m];
do
{
while(x[down] < pivot && down <n)
{
down++;
}
while(x[up] > pivot && up>m)
{
up--;
}
if(down < up)
{

temp=x[down];
x[down]=x[up];
x[up]=temp;
}
}while(down< up);
temp=x[m];
x[m]=x[up];
x[up]=temp;
Qsort(m,up-1);
Qsort(up+1,n);
}
}
```

Q.2) Write a C program to implement dynamic stack of integer with operations: (• Push () • Pop () • Search ()

```
#include<stdio.h>
#define NODEALLOC (struct node*) malloc(sizeof(struct node))

typedef struct node
{
    char data;
    struct node *next;
}Stack;

Stack *top;

void initstack()
{
    top=NULL;

}

int isempty()
{
  return (top==NULL);
}


void push(char n)
{
   Stack *newnode;
   newnode = NODEALLOC;
   newnode->data=n;
   newnode->next=top;
   top=newnode;
}

char pop()
{
   char num;
   Stack *temp=top;
   num=top->data;
   top=top->next;
   free(temp);
   return num;
}

void show()
 {
```

```
           Stack *temp;
           printf(" \n Elements in Stack \n");
           temp=top;
           do
           {
            printf("\n %d",temp->data);
            temp=temp->next;
           } while(temp!=NULL);

}



int search()
 {

           Stack *temp;
           int key,flag =0;
           printf(" \n Enter elements to search \n");
           scanf(“%d”,&key);
          temp=top;
           do
           {
            if(temp->data==key)
              flag=1 ;
            else
            temp=temp->next;
           } while(temp!=NULL);


return flag;

}



#include<string.h>

main()
{

   int i=0,n,n1,ch;
   initstack();
     do
        {
           printf("\n--------MENU---------------------------");
           printf("\n 1) Push element");
           printf("\n 2) POP element");
           printf("\n 3) Show element");
           printf("\n 4) Search");
           printf(“\n 5)Exit);
           printf("\n Enter your choice ");
           scanf("%d",&ch);
           switch(ch)
           {
            case 1:
               printf("\n Enter the element to insert :  ");
               scanf("%d",&n);
               push(n);
               break;



             case 2:
               if(!isempty())
               {
               n1=pop();
               printf("\n %d is deleted from stack",n1);
               }
               else
               {
                   printf("\n Stack is empty \n");
               }
               break;

            case 3:
               printf("\n The element in stack are as follows \n");
               show();
               break;

               case 4:
               n1=search();

               if(n1==1)
                   printf(“\n\n %d is FOUND in  Stack”,n1);
               else
                   printf(“\n\n %d is NOT FOUND in  Stack”,n1);


               break;
           }
        }while(ch!=4);
     return;
 }
```

Slip-14

Q.1) Write a C program to search a given character using binary search method [use recursion]

```
#include <stdio.h>

// Recursive binary search function
int binarySearch(char arr[], int low, int high, char key) {
    if (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == key) {
            return mid;  // Character found, return index
        }
        else if (arr[mid] < key) {
            return binarySearch(arr, mid + 1, high, key);  // Search
in right half
        }
        else {
            return binarySearch(arr, low, mid - 1, key);   // Search
in left half
        }
    }
    return -1; // Character not found
}

int main() {
    char arr[100], key;
    int n, result;

    printf("Enter number of characters: ");
    scanf("%d", &n);

    printf("Enter characters in sorted order: ");
    for (int i = 0; i < n; i++) {
        scanf(" %c", &arr[i]); // Note the space before %c
    }

    printf("Enter character to search: ");
    scanf(" %c", &key);

    result = binarySearch(arr, 0, n - 1, key);

    if (result != -1)
        printf("Character '%c' found at position %d\n", key, result
+ 1);
    else
        printf("Character '%c' not found in the array.\n", key);

    return 0;
}
```

Q.2) Write a C program to create BST and implement following operations: ● Display in-order traversal ● To count total no of nodes ● To count odd numbers from BST

```
#include<stdio.h>
#define NODEALLOC (struct node *) malloc (sizeof (struct node ))
typedef struct node
{
int data;
struct Node *left,*right;
}NODE;
// ------function for creating BST------
NODE * createbst(NODE *root)
{
NODE *newnode,*temp;
int num,n,i;
printf("\n Enter how many node \n");
scanf("%d",&n);
for(i=1;i<=n;++i)
{
newnode=NODEALLOC;
printf(" \n Enter the data to insert ");
scanf("%d",&num);
newnode->data=num;
newnode->left=NULL;
newnode->right=NULL;
if(root==NULL)
root=newnode;
else
{
temp=root;
while(1)
{
if(num < temp->data)
{
if(temp->left==NULL) // not left child
{
temp->left=newnode;
break;
}
else
temp=temp->left; // move left
}
else
if(num > temp->data)
{
if(temp->right == NULL)
{
temp->right=newnode;
break;
}
else

temp=temp->right;
}
} // end while
}
} // for end
return root;
}
```

```
void inorder(NODE * root)
{
NODE *temp=root;
if(temp!=NULL)
{
inorder(temp->left);
printf("%d ",temp->data);
inorder(temp->right);
}
}


int countnodes(NODE *root)
{
static int count =0;
NODE *temp =root;
if(temp!=NULL)
{
count++;
countnodes(temp->left);
countnodes(temp->right);
}
return count;
}


int countOdd(NODE * root)
 {

     int count =0;
    if (root == NULL)
       return 0;
    count = (root->data % 2 != 0) ? 1 : 0;
    count += countOdd(root->left);
    count += countOdd(root->right);
    return count;
}
//----------------------------------------------------------------

//---------------------------main fun()---------------------
int main()
{
NODE *root=NULL;
int count,totalnode;
root= createbst(root);
printf("\n Node in Inorder \n\n");
inorder(root);


printf("\n Total Node in Binary Tree : ");
totalnode=countnodes(root);
printf("%d",totalnode);

printf("\n Count ODD Numbers in Binary Tree : ");
totalnode=countOdd(root);
printf("%d",totalnode);
return 0;
}
```

Slip 15

Q1. Write a C program to sort n numbers using Quick sort.

```
#include<stdio.h>

int x[20];
void main()
{
int n,i;
printf("\n\n--------------Quick Sort--------------------\n\n");
printf("\n Enter how many number u want to enter :----> ");
scanf("%d",&n);
printf("\n\n Enter all array element\n\n");
```

```
for(i=0;i<n;++i)
{
scanf("%d",&x[i]);
}
Qsort(0,n-1);
printf("\n\n--------------------The Sorted array -------------
\n\n");
for(i=0;i<n;++i)
{
printf("%d ",x[i]);
}
}
//-----------Function for Quick sort----------------------
Qsort(int m,int n)
{
int down,up,temp,pivot;
if(m<n)
{
down=m+1;
up=n;
pivot=x[m];
do
{
while(x[down] < pivot && down <n)
{
down++;
}
while(x[up] > pivot && up>m)
{
up--;
}
if(down < up)

{

temp=x[down];
x[down]=x[up];
x[up]=temp;
}
}while(down< up);
temp=x[m];
x[m]=x[up];
x[up]=temp;
Qsort(m,up-1);
Qsort(up+1,n);
}
```

```
}
```

Q.2) Write a C program to reverse a given string using stack.

```
// Program for reverse a string using stack.
#include<stdio.h>
#define MAX 5
#define EMPTY -1
#define FULL MAX-1
typedef struct stk
{
int top;
int item[MAX];
}stack;
// -------------------initstack------------------------------
void initstack(stack *ps)
{
```

```
ps->top=-1;
}
//---------------------------------------------------------------
int isempty(stack *ps)
{
return(ps->top==EMPTY);
}
//--------------------------------------------------------------

int isfull(stack *ps)
{
return (ps->top==FULL);
}
//----------------------------------------------------------------
void push( stack *ps,int n)
{
if(isfull(ps))
{
printf("\n stack full");
}
else
{
++ps->top;
ps->item[ps->top]=n;
}
}
//-------------------------------------------------------------------
-i
int pop(stack *ps)
{ int n1;
n1=ps->item[ps->top];
ps->top-- ;
return n1;
}

//-------------------------------------------------------------------
----
void show(stack *ps)
{

int t;
t=ps->top;
while(t>=0)
{
printf("%d ",ps->item[t]);
t--;
}
}
int main()
{
stack s1;
char str[20];
int i=0;
initstack(&s1);
printf("\n enter the string\n");
gets(str);
//scanf("%s",str);
while(str[i]!='\0')
{
push(&s1,str[i]);
i++;
}
i=0;
while(!isempty(&s1))
{
str[i]=pop(&s1);
i++;
}
str[i]='\0';
printf("\n The reversed of string is :");
printf("%s\n",str);
return 0;
}
```

Slip 16

Q.1) Write a C program to sort character array using bubble sort.

```
#include<stdio.h>

//  function for bubble sort

Bubblesort(char x[],int n)
  {
      int t,i,j;

      for(i=0;i<n;i++)
      {

         for(j=0;j<n-1;j++)
         {

              if(x[j] > x[j+1])
              {

                  t      = x[j];
                  x[j]   = x[j+1];
                  x[j+1] = t;
              }  //if end


          }  //j end
       } //i end

 }
//-------------------------------------------------------------

  int  main()
  {
      char x[20],n,i;

      printf("\n\n-----------Bubble Sort--------------------\n\n");
      printf("\n Enter how many character u want to enter :----> ");
      scanf("%d",&n);
      printf("\n\n Enter all character\n\n");
      for(i=0;i<n;++i)
      {
          scanf("%c",&x[i]);
      }
      Bubblesort(x,n);
      printf("\n\n----------The Sorted array --------------\n\n");
      for(i=0;i<n;++i)
      {
          printf("%dc ",x[i]);
      }

  return;

  }
```

Q.2) Write a C program to convert infix expression into Postfix.

```
#include<stdio.h>

  #define MAX 5
  #define EMPTY -1
  #define FULL MAX-1

  typedef struct stk
  {
        int top;
        int item[MAX];
  }stack;

  // -------------------initstack------------------------------
```

```
void initstack(stack *ps)
{
      ps->top=-1;
}
//--------------------------------------------------------------
int isempty(stack *ps)
{
        return(ps->top==EMPTY);
}
//-------------------------------------------------------------

int isfull(stack *ps)
{
       return (ps->top==FULL);
}
//---------------------------------------------------------------

void push( stack *ps,int n)
{
      if(isfull(ps))
      {
       printf("\n stack full");
      }
      else
      {
      ++ps->top;
      ps->item[ps->top]=n;
      }
}

//-----------------------------------------------------------

int pop(stack *ps)
{   int n1;
       n1=ps->item[ps->top];
       ps->top-- ;
       return n1;
}

//--------Conversion of infix to postfix ---------------------

void postfix(char in[],char post[])
{
     int i,j=0;
     char ch;
     stack s1;
     initstack(&s1);
     for(i=0;in[i]!='\0';i++)
     {
        if(isalpha(in[i]))
         {
            post[j]=in[i];
            j++;
         }
         else
         {
           switch(in[i])
            {
      case '+' :
      case '-' :
      case '*' :
      case '/' :
      case '%' :
      case '(' :
             push(&s1,in[i]);
             break;
      case ')' :
             while((ch=pop(&s1))!='(')
             {
                    post[j]=ch;
                    j++;
             }
             }
           }
        }
       while(!isempty(&s1))
       {
             post[j]=pop(&s1);
             j++;
       }
       post[j]='\0';
     }

//----------------------main-----------------------------

void main()
{

       char in[20],post[20];

       printf("\n Enter Infix string : ");
       scanf("%s",in);
       fflush(stdin);
       postfix(in,post); //--------------- postfix conversions
       printf(" \n Postfix sring is ");
       printf("%s",post);
      }
```

Slip 18

Q.1) Write a C program to display the city code of the corresponding city name using linear search method. The structure is: struct city { int city_code; char name[30]; }

```
struct city
{
char cname[10];
int STD;
}C[10];
int Linearsearch(struct city A[], int last, char target[], int
*location)
{
int i;
i=0;
while (i<last && strcmp(target, A[i].cname)!=0)
i++;
*location = i;
return (strcmp(target, A[i].cname) );
}
//-----------------------------------------------------------
int main(void)
{
int arr[10];
int x ,result,n,index,i,num,t2;
char fname[10],line[50],t1[20],name[10];
FILE *fp;
printf("\n Enter Filename : ");
scanf("%s",fname);
if((fp=fopen(fname,"r"))==NULL)
{
printf("\n Error in opening File \n");
return;
}
fflush(stdin);
n=0;
printf("\n------------- File Data------------\n");
while(fgets(line,80,fp))
{
sscanf(line,"%s %d",t1,&t2);
strcpy(C[n].cname,t1);
C[n].STD=t2;
printf("\n\t %s\t %d",C[n].cname,C[n].STD);
n++;
}
printf("\n Enter cityname to search " );
scanf("%s",name);
result = Linearsearch(C, n, name,&index);
if(!result)
printf("\n Element is present at---> %d location and \n STD Code is
--
---> %d", index +1,C[index].STD);

else
printf("\n Element is not present");
return 0;
}
```

Q.2) Write a C program to implement dynamic implementation of queue with following operations: • Insert • Length-Count total elements • Search-Search particular element

```
#include<stdio.h>
typedef struct node
{
int data;
struct node *next;
}NODE;
NODE *front,*rear;
//------------------------------------------------------------
void initQ()
```

```
{
front=rear=NULL;
}
//------------------------------------------------------------
void addQ(int n)
{
NODE *newnode;
newnode=( NODE *)malloc(sizeof (NODE));
newnode->data=n;
newnode->next=NULL;
if(front==NULL)
{
rear=front=newnode;
}
else
{
rear->next=newnode;
rear=newnode;
}
}

//--------------------------------------------------------
int isempty()

{
return (front==NULL);
}
//------------------------------------------------------------
void display()
{
NODE *temp;
temp=front;
printf("\n Elemnt");
do
{
printf("\n %d",temp->data);
temp=temp->next;
}while(temp!=NULL);
}

//-----------------------------------

void Search()
{
    int key,flag =0 ;
NODE *temp;
```

```
temp=front;
printf("\n Enter Element to search ");
scanf("%d",&key);

do
{
 if(temp->data==key)
  { flag=1;       break;}
 else
    temp=temp->next;
}while(temp!=NULL);

if (flag==1)
    printf("\n\n %d is present in Queue");
else
   printf("\n\n %d isNOT present in Queue");

}
//---------------------------------------------
void Length()
{
     int count;
NODE *temp;
temp=front;
printf("\n Count Number of elements in Q : ");
do
{
count++;
temp=temp->next;
}while(temp!=NULL);

printf("%d",count );
}
//---------------------------------------------
int main()
{
int ch,n;
initQ();
do
{
printf("\n 1.Insert Node \n 2.Serach  Node\n 3.Length - count total
element \n 4.Display\n 5.exit");
printf("\n Enter ur choice : ");
scanf("%d",&ch);
switch(ch)
{
```

```
case 1:
printf("\n Enter element : ");
scanf("%d",&n);
addQ(n);
break;
case 2:
    Search();
break;

case 3:
    Length();
break;
case 4:
display();
break;
case 5:
break;

}
}while(ch!=5);
}
```

<u>Slip 19</u>

<u>Q.1) Write a C program to create and display singly Linked List of vowels.</u>

```
 #include<stdio.h>

  typedef struct node
  {
     char data;
     struct node *next;
  }NODE;
```

```
//------------create function()------------------------

void createlist(NODE *head)
{
   int n ,i;
   NODE *last,*newnode;

   printf("\n Enter how many nodes :  ");
   scanf("%d",&n);

   last = head;

   for(i=1;i<=n;i++)
   {

       newnode  =  (NODE *) malloc (sizeof(NODE));
       newnode->next=NULL;

       printf("\n enter the node data  : ");

       scanf(" %c",&newnode->data);

       last->next = newnode;

       last=newnode;
  }

 }


//-------------display()--------------------


void display(NODE *head)
{
  NODE *temp;

  for(temp=head->next;temp!=NULL;temp=temp->next)
  {
     printf("%c\t",temp->data);
  }

}
```

```
void main()
{
   NODE *head;
   int ch,n,pos;

   head=(NODE *) malloc (sizeof(NODE));




              createlist(head);

             printf("\n\n  Singly Linked List All Elements \n\n");
             display(head);
}
```

Q.2) Write a C program to implement dynamic implementation of stack with following operations: ● Push ● Pop ● Reverse-Display elements in reverse order of insertion

```
#include<stdio.h>
#define NODEALLOC (struct node*) malloc(sizeof(struct node))

typedef struct node
{
    char data;
    struct node *next;
}Stack;

Stack *top;
```

```
void initstack()
{
    top=NULL;

}

int isempty()
{
  return (top==NULL);
}


void push(char n)
{
   Stack *newnode;
   newnode = NODEALLOC;
   newnode->data=n;
   newnode->next=top;
   top=newnode;
}

char pop()
{
   char num;
   Stack *temp=top;
   num=top->data;
   top=top->next;
   free(temp);
   return num;
}

void show()
 {

           Stack *temp;
           printf(" \n Elements in Stack \n");
           temp=top;
           do
           {
            printf("\n %d",temp->data);
            temp=temp->next;
           } while(temp!=NULL);

}

#include<string.h>

main()
```

```
{

   int i=0,n,n1,ch;
   initstack();
     do
        {
           printf("\n--------------------MENU-----------------------
----");
           printf("\n 1) Push element");
           printf("\n 2) POP element");
           printf("\n 3) Show element");
           printf("\n 4) Exit");
           printf("\n Enter your choice ");
           scanf("%d",&ch);
           switch(ch)
           {
            case 1:
               printf("\n Enter the element to insert :  ");
               scanf("%d",&n);
               push(n);
               break;



             case 2:
               if(!isempty())
               {
               n1=pop();
               printf("\n %d is deleted from stack",n1);
               }
               else
               {
                   printf("\n Stack is empty \n");
               }
               break;

            case 3:
               printf("\n The element in stack are as follows \n");
               show();
               break;

               case 4:
               break;
           }
        }while(ch!=4);
     return;
 }
```

Slip 20

Q.1) Write a C program to sort an array using insertion sort method.

```
#include<stdio.h>
Insertsort(int x[],int n)
{
int next,i,newelement;
for(next=1;next<n;++next)
{
//newelement is element to be inserted
newelement=x[next];
// shift element > newelement to right by 1 pos
for(i=next-1;i>=0 && newelement<x[i];i--)
{
x[i+1]=x[i];
}
//insert new element at pos i+1
x[i+1]=newelement;
}
}
//-----------------------------------------------------------

int main()
{
int x[20],n,i;
printf("\n\n-------------Insertion Sort--------------------\n\n");
printf("\n Enter how many number u want to enter :----> ");
scanf("%d",&n);
printf("\n\n Enter all array element\n\n");
for(i=0;i<n;++i)
{

scanf("%d",&x[i]);
}
Insertsort(x,n);
printf("\n\n--------------------The Sorted array -------------
\n\n");
for(i=0;i<n;++i)
{
printf("%d ",x[i]);
}
}
```

Q.2) Write a C program to convert an infix expression to a postfix expression.

```
#include<stdio.h>

  #define MAX 5
  #define EMPTY -1
  #define FULL MAX-1

  typedef struct stk
  {
        int top;
        int item[MAX];
```

```
}stack;

// -------------------initstack------------------------------

void initstack(stack *ps)
{
      ps->top=-1;
}
//---------------------------------------------------------------
int isempty(stack *ps)
{
        return(ps->top==EMPTY);
}
//--------------------------------------------------------------

int isfull(stack *ps)
{
       return (ps->top==FULL);
}
//----------------------------------------------------------------

void push( stack *ps,int n)
{
      if(isfull(ps))
      {
       printf("\n stack full");
      }
      else
      {
      ++ps->top;
      ps->item[ps->top]=n;
      }
}

//-----------------------------------------------------------

int pop(stack *ps)
{   int n1;
       n1=ps->item[ps->top];
       ps->top-- ;
       return n1;
}

//--------Conversion of infix to postfix ---------------------

void postfix(char in[],char post[])
{
     int i,j=0;
     char ch;
```

```
      stack s1;
      initstack(&s1);
      for(i=0;in[i]!='\0';i++)
      {

         if(isalpha(in[i]))
          {
             post[j]=in[i];
             j++;
          }
          else
          {
            switch(in[i])
             {
       case '+' :
       case '-' :
       case '*' :
       case '/' :
       case '%' :
       case '(' :
             push(&s1,in[i]);
             break;
       case ')' :
             while((ch=pop(&s1))!='(')
             {
                    post[j]=ch;
                    j++;
             }
             }
           }
         }
        while(!isempty(&s1))
        {
             post[j]=pop(&s1);
             j++;
        }
        post[j]='\0';
      }

 //-----------------------main--------------------------------

 void main()
 {

        char in[20],post[20];

        printf("\n Enter Infix string : ");
        scanf("%s",in);
        fflush(stdin);
```

```
         postfix(in,post); //--------------- postfix conversions
         printf(" \n Postfix sring is ");
         printf("%s",post);
        }
```

Slip 07

Q.1) Write a C program to find the length of singly linked list

```
#include<stdio.h>

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

```
//-----------create function()----------------------

void createlist(NODE *head)
{
   int n ,i;
   char ch;
   NODE *last,*newnode;

     last = head;

   do
   {

       newnode  =  (NODE *) malloc (sizeof(NODE));
       newnode->next=NULL;

       printf("\n Enter the node data to Insert in Linked List  :
");

       scanf(" %d",&newnode->data);

       last->next = newnode;

       last=newnode;

       while ((getchar()) != '\n');

        printf("Do you want to insert another node (y/n)? ");
        scanf("%c", &ch);

    } while (ch == 'y' || ch == 'Y');
 }


//-------------display()-------------------


void display(NODE *head)
{
  NODE *temp;

  for(temp=head->next;temp!=NULL;temp=temp->next)
  {
     printf("%d\t",temp->data);
  }
}
//----------------------------------------------------------

void Length(NODE *head)
{
  NODE *temp;
  int count=0;

  for(temp=head->next;temp!=NULL;temp=temp->next)
  {
    count ++;
  }

printf("\n \n Length of Linked List  :  %d",count);
}
//-------------------------------------------------------------------
void main()
{
   NODE *head;
   int ch,n,pos;

   head=(NODE *) malloc (sizeof(NODE));
    createlist(head);

      printf("\n\n  Singly Linked List All Elements \n\n");
                 display(head);
                 Length(head);
}
```

Q.2) Write a C program to implement dynamic implementation of stack of integers with following operation: • push() • pop () • isempty() • isfull() • display ()

```
#include<stdio.h>
#define NODEALLOC (struct node*) malloc(sizeof(struct node))

typedef struct node
{
    char data;
    struct node *next;
}Stack;

Stack *top;

void initstack()
{
    top=NULL;

}

int isempty()
{
  return (top==NULL);
}


void push(char n)
{
   Stack *newnode;
   newnode = NODEALLOC;
   newnode->data=n;
   newnode->next=top;
   top=newnode;
}

char pop()
{
   char num;
   Stack *temp=top;
   num=top->data;
   top=top->next;
   free(temp);
   return num;
}
```

```
void show()
 {

           Stack *temp;
           printf(" \n Elements in Stack \n");
           temp=top;
           do
           {
            printf("\n %d",temp->data);
            temp=temp->next;
           } while(temp!=NULL);

}

#include<string.h>

main()
{

   int i=0,n,n1,ch;
   initstack();
     do
        {
           printf("\n-----------MENU---------------------------");
           printf("\n 1) Push element");
           printf("\n 2) POP element");
           printf("\n 3) Show element");
           printf("\n 4) Exit");
           printf("\n Enter your choice ");
           scanf("%d",&ch);
           switch(ch)
           {
            case 1:
               printf("\n Enter the element to insert :  ");
               scanf("%d",&n);
               push(n);
               break;
           case 2:
               if(!isempty())
               {
               n1=pop();
               printf("\n %d is deleted from stack",n1);
               }
               else
               {
                   printf("\n Stack is empty \n");
               }
               break;
```

```
                case 3:
                   printf("\n The element in stack are as follows \n");
                   show();
                   break;

                   case 4:
                   break;
               }
            }while(ch!=4);
        return;
    }
```