Bengaluru North University

II Sem BCA (NEP)

Data Structure Using C Programs

Part A:

1. Program to find GCD using recursive function
2. Program to display Pascal Triangle using binomial function
3. Program to generate n Fibonacci numbers using recursive function.
4. Program to implement Towers of Hanoi.
5. Program to implement dynamic array, find smallest and largest element of the array.
6. Program to create two files to store even and odd numbers.
7. Program to create a file to store student records.
8. Program to read the names of cities and arrange them alphabetically.
9. Program to sort the given list using selection sort technique.
10. Program to sort the given list using bubble sort technique.

Part B:

1. Program to sort the given list using insertion sort technique.
2. Program to sort the given list using quick sort technique.
3. Program to sort the given list using merge sort technique.
4. Program to search an element using linear search technique.
5. Program to search an element using recursive binary search technique.
6. Program to implement Stack.
7. Program to convert an infix expression to postfix.
8. Program to implement simple queue.
9. Program to implement linear linked list.
10. Program to display traversal of a tree

// C Program To Find GCD of Two Number Using Recursion

```
#include <stdio.h>
int GCD(int x, int y);
int main()
{
   int a, b;
      // Asking for Input
   printf("Enter Two Positive Integers: \n");
   scanf("%d\n %d", &a, &b);
      printf("GCD of %d and %d is %d.", a, b, GCD(a, b));
   return 0;
}

int GCD(int x, int y)
{
   if( y != 0)
      return GCD(y, x % y);
   else
      return x;
}
```

```
Enter two positive integers: 366
60
G.C.D of 366 and 60 is 6.
```

// C Program to Print Pascal Triangle

```
#include < stdio.h >
long factorial(int);
int main()
{
   int i, n, c;
   printf("Enter the number of rows you wish to see in pascal triangle\n");
   scanf("%d", & n);
   for (i = 0; i < n; i++) {
      for (c = 0; c <= (n - i - 2); c++)
         printf(" ");
      for (c = 0; c <= i; c++)
         printf("%ld ", factorial(i) / (factorial(c) * factorial(i - c)));
      printf("\n");
   }
   return 0;
}
long factorial(int n) {
   int c;
   long result = 1;
   for (c = 1; c <= n; c++)
         result = result * c;
   return result;
}
```

```
Enter the number of rows you wish to see in pascal triangle
5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
Enter the number of rows you wish to see in pascal triangle
```

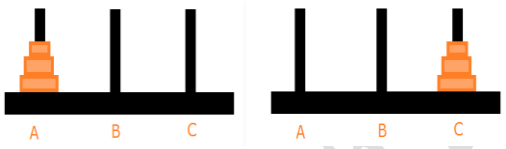
// C Program to Compute fibonacci numbers using recursion method

```
#include<stdio.h>
 int Fibonacci(int);
 int main()
{
   int n, i = 0, c;
printf("Enter total terms\n");
scanf("%d",&n);
   printf("Fibonacci series\n");
    for ( c = 1 ; c <= n ; c++ )
   {
      printf("%d\n", Fibonacci(i));
      i++;
   }
    return 0;
}
int Fibonacci(int n)
{
   if ( n == 0 )
      return 0;
   else if ( n == 1 )
      return 1;
   else
      return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

```
Enter Total terms:
6
Fibonacci series terms are:
0
1
1
2
3
5
```

//C program for Tower of Hanoi using Recursion

```
#include <stdio.h>
 void towers(int, char, char, char);
 int main()
{
    int num;
     printf("Enter the number of disks : ");
    scanf("%d", &num);
    printf("The sequence of moves involved in the Tower of Hanoi are :\n");
    towers(num, 'A', 'C', 'B');
    return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
    if (num == 1)
    {
        printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
        return;
    }
    towers(num - 1, frompeg, auxpeg, topeg);
    printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
    towers(num - 1, auxpeg, topeg, frompeg);
}
```




```
Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

Move disk 1 from peg A to peg C
Move disk 2 from peg A to peg B
Move disk 1 from peg C to peg B
Move disk 3 from peg A to peg C
Move disk 1 from peg B to peg A
Move disk 2 from peg B to peg C
Move disk 1 from peg A to peg C
```

//C program to find the largest and smallest element in an array

```
#include<stdio.h>

int main()
{
int a[50],i,n,large,small;
printf("How many elements:");
scanf("%d",&n);
printf("Enter the Array:");

for(i=0;i<n;++i)
scanf("%d",&a[i]);
large=small=a[0];
for(i=1;i<n;++i)
{
if(a[i]>large)
large=a[i];
if(a[i]<small)
small=a[i];
}
printf("The largest element is %d",large);
printf("\nThe smallest element is %d",small);

return 0;
}
```

Output:
How many elements:5
Enter the Array:1 8 12 4 6
The largest element is 12
The smallest element is 1

//C Program to Write Odd and Even Numbers into Different Files

```
#include<stdio.h>
#include<conio.h>
void main()
{
FILE *fp,*fp1,*fp2;
int c,i;
clrscr();
fp=fopen("data.txt","w");
printf("Enter the numbers");
for(i=0;i<10;i++)
{
        scanf("%d",&c);
        putw(c,fp);
}
fclose(fp);
fp=fopen("data.txt","r");
fp1=fopen("even.txt","w");
fp2=fopen("odd.txt","w");
while((c=getw(fp))!=EOF)
{
if(c%2==0)
        putw(c,fp1);
else
        putw(c,fp2);
}
fclose(fp);
fclose(fp1);
fclose(fp2);
fp1=fopen("even.txt","r");
while((c=getw(fp1))!=EOF)
        printf("File-1 Elements : %4d",c);
        printf("\n\n");
fp2=fopen("odd.txt","r");
while((c=getw(fp2))!=EOF)
        printf("File-2 Elements : %4d",c);
        fcloseall();
}
```

Output:

Enter the numbers: 1,2,3,4,5,6,7,8,9,10

File-1 Elements: 2 4 6 8 10
File-2 Elements: 1 3 5 7 9

Write a C program to store record of Student Details in **a file using structure.**

```
#include<stdio.h>
struct stud
{
    int rno;
    float per;
    char name[20],add[20];
}s;
int main()
{
    FILE *fp;
        fp=fopen("student.txt","w");
            printf("Enter record of student:\n\n");
            printf("\nEnter student number : ");
            scanf("%d",&s.rno);
            printf("\nEnter name of student: ");
            scanf("%s",s.name);
            printf("\nEnter student address : ");
            scanf("%s",s.add);
            printf("\nEnter percentage of student : ");
            scanf("%f",&s.per);
            fprintf(fp,"%d\n%s\n%s\n%f",s.rno,s.name,s.add,s.per);
            printf("\nRecord stored in file...");
        fclose(fp);
    return 0;
}
```

Output:

```
Enter student number : 1

Enter name of student: ADISESHA

Enter student address : SJES_COLLEGE

Enter percentage of student : 75

Record stored in file...
```

//C program to sort city names in alphabetical order

```
#include<stdio.h>
#include<string.h>
main(){
  int i,j,n;
  char str[100][100],s[100];
  printf("Enter number of City names :\n");
  scanf("%d",&n);
  printf("Enter City names in any order:\n");
  for(i=0;i<n;i++){
    scanf("%s",str[i]);
  }
  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){
      if(strcmp(str[i],str[j])>0){
        strcpy(s,str[i]);
        strcpy(str[i],str[j]);
        strcpy(str[j],s);
      }
    }
  }
printf("\nThe sorted order of names are:\n");
  for(i=0;i<n;i++){
    printf("%s\n",str[i]);
  }
}
```

Output

Enter number of City names:5

Enter City names in any order:
Mumbai
Chennai
Delhi
Bangalore
Agra

The sorted order of names are:
Agra
Bangalore
Chennai
Delhi
Mumbai

C program to sort the given list using selection sort technique

```
#include<stdio.h>
int main()
{
  int i, j, count, temp, number[25];
  printf("How many numbers u are going to enter?: ");
  scanf("%d",&count);
  printf("Enter %d elements: ", count);
  // Loop to get the elements stored in array
  for(i=0;i<count;i++)
    scanf("%d",&number[i]);
   // Logic of selection sort algorithm
  for(i=0;i<count;i++){
    for(j=i+1;j<count;j++){
      if(number[i]>number[j]){
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }
  }
  printf("Sorted elements: ");
  for(i=0;i<count;i++)
    printf(" %d",number[i]);
  return 0;
}
```

//C program to sort the given list using bubble sort technique

```
#include <stdio.h>
int main()
{
  int array[100], n, c, d, swap;
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  for (c = 0 ; c < n - 1; c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use '<' instead of '>' */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
  printf("Sorted list in ascending order:\n");
  for (c = 0; c < n; c++)
     printf("%d\n", array[c]);
  return 0;
}
```

// C Program to sort an array in ascending order using Insertion Sort

```
#include <stdio.h>
int main()
{
   int n, i, j, temp;
   int arr[64];
    printf("Enter number of elements\n");
   scanf("%d", &n);
    printf("Enter %d integers\n", n);
   for (i = 0; i < n; i++)
   {
      scanf("%d", &arr[i]);
   }
   for (i = 1 ; i <= n - 1; i++)
   {
            j = i;
         while ( j > 0 && arr[j-1] > arr[j])
         {
            temp     = arr[j];
            arr[j]   = arr[j-1];
            arr[j-1] = temp;
            j--;
         }
   }
   printf("Sorted list in ascending order:\n");
   for (i = 0; i <= n - 1; i++)
   {
      printf("%d\n", arr[i]);
   }
   return 0;
}
```

// C Program to Perform Quick Sort on a set of Entries from a File using Recursion

```
#include <stdio.h>
void quicksort (int [], int, int);
int main()
{
   int list[50];
   int size, i;

   printf("Enter the number of elements: ");
   scanf("%d", &size);
   printf("Enter the elements to be sorted:\n");
   for (i = 0; i < size; i++)
   {
      scanf("%d", &list[i]);
   }
   quicksort(list, 0, size - 1);
   printf("After applying quick sort\n");
   for (i = 0; i < size; i++)
   {
      printf("%d ", list[i]);
   }
   printf("\n");

   return 0;
}
void quicksort(int list[], int low, int high)
{
   int pivot, i, j, temp;
   if (low < high)
   {
      pivot = low;
      i = low;
      j = high;
      while (i < j)
      {
         while (list[i] <= list[pivot] && i <= high)
         {
            i++;
         }
         while (list[j] > list[pivot] && j >= low)
         {
            j--;
         }
         if (i < j)
         {
            temp = list[i];
```

```
            list[i] = list[j];
            list[j] = temp;
        }
      }
      temp = list[j];
      list[j] = list[pivot];
      list[pivot] = temp;
      quicksort(list, low, j - 1);
      quicksort(list, j + 1, high);
   }
}
```

//C Program to Input Few Numbers & Perform Merge Sort on them using Recursion

```
#include <stdio.h>
void mergeSort(int [], int, int, int);
void partition(int [],int, int);
int main()
{
   int list[50];
   int i, size;

   printf("Enter total number of elements:");
   scanf("%d", &size);
   printf("Enter the elements:\n");
   for(i = 0; i < size; i++)
   {
       scanf("%d", &list[i]);
   }
   partition(list, 0, size - 1);
   printf("After merge sort:\n");
   for(i = 0;i < size; i++)
   {
       printf("%d   ",list[i]);
   }

  return 0;
}

void partition(int list[],int low,int high)
{
   int mid;

   if(low < high)
   {
      mid = (low + high) / 2;
      partition(list, low, mid);
      partition(list, mid + 1, high);
      mergeSort(list, low, mid, high);
   }
}

void mergeSort(int list[],int low,int mid,int high)
{
   int i, mi, k, lo, temp[50];
    lo = low;
   i = low;
   mi = mid + 1;
   while ((lo <= mid) && (mi <= high))
```

```
    {
        if (list[lo] <= list[mi])
        {
            temp[i] = list[lo];
            lo++;
        }
        else
        {
            temp[i] = list[mi];
            mi++;
        }
        i++;
    }
    if (lo > mid)
    {
        for (k = mi; k <= high; k++)
        {
            temp[i] = list[k];
            i++;
        }
    }
    else
    {
        for (k = lo; k <= mid; k++)
        {
             temp[i] = list[k];
             i++;
        }
    }
     for (k = low; k <= high; k++)
    {
        list[k] = temp[k];
    }
}
```

// C Program to implement Linear Search Algorithm recursively

```
#include<stdio.h>
int Linear_search(int arr[], int Search_ele, int n)
{
   int i;
   static int temp=0;
   if(n>0)
   {
      i=n-1;
        if(arr[i]==Search_ele)
          temp=1;
      Linear_search(arr,Search_ele,i);
   }
return temp;
}
int main()
{
   int n,j;
   printf("Enter your array size:");
   scanf("%d",&n);
   int arr[n];
   printf("Enter the Array Element:");
   for(j=0;j<n;j++)
   {
      scanf("%d",&arr[j]);
   }
   int Search_ele;
   printf("Enter the search element:");
   scanf("%d",&Search_ele);
   if(Linear_search(arr,Search_ele,n)==1)
      printf("Element found....");
   else
      printf("Element not found....");
   return 0;
}
```

//C Program to Perform Binary Search using Recursion

```
#include<stdio.h>
#include<stdlib.h>
int binsearch(int[], int, int, int);

int main() {
   int num, i, key, position;
   int low, high, list[10];
    printf("\nEnter the total number of elements");
   scanf("%d", &num);
   printf("\nEnter the elements of list :");
   for (i = 0; i < num; i++) {
      scanf("%d", &list[i]);
   }
   low = 0;
   high = num - 1;
   printf("\nEnter element to be searched : ");
   scanf("%d", &key);
   position = binsearch(list, key, low, high);
   if (position != -1) {
      printf("\nNumber present at %d", (position + 1));
   }
else
      printf("\n The number is not present in the list");
   return (0);
}

// Binary search function for binary search
int binsearch(int a[], int x, int low, int high) {
   int mid;
   if (low > high)
      return -1;
   mid = (low + high) / 2;
   if (x == a[mid]) {
      return (mid);
   }
else if (x < a[mid]) {
      binsearch(a, x, low, mid - 1);
   }
else {
      binsearch(a, x, mid + 1, high);
   }
}
```

// C program to implement stack. Stack is a LIFO data structure.

```
#include <stdio.h>
#define MAXSIZE 5
struct stack
{
   int stk[MAXSIZE];
   int top;
};
typedef struct stack STACK;
STACK s;

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

void main ()
{
   int choice;
   int option = 1;
   s.top = -1;

   printf ("STACK OPERATION\n");
   while (option)
   {
      printf ("-----------------------------------------\n");
      printf ("      1    -->    PUSH          \n");
      printf ("      2    -->    POP           \n");
      printf ("      3    -->    DISPLAY           \n");
      printf ("      4    -->    EXIT        \n");
      printf ("-----------------------------------------\n");

      printf ("Enter your choice\n");
      scanf    ("%d", &choice);
      switch (choice)
      {
      case 1:
         push();
         break;
      case 2:
         pop();
         break;
      case 3:
         display();
         break;
      case 4:
         return;
```

```
        }
        fflush (stdin);
        printf ("Do you want to continue(Type 0 or 1)?\n");
        scanf    ("%d", &option);
    }
}
/*  Function to add an element to the stack */
void push ()
{
    int num;
    if (s.top == (MAXSIZE - 1))
    {
        printf ("Stack is Full\n");
        return;
    }
    else
    {
        printf ("Enter the element to be pushed\n");
        scanf ("%d", &num);
        s.top = s.top + 1;
        s.stk[s.top] = num;
    }
    return;
}
/*  Function to delete an element from the stack */
int pop ()
{
    int num;
    if (s.top == - 1)
    {
        printf ("Stack is Empty\n");
        return (s.top);
    }
    else
    {
        num = s.stk[s.top];
        printf ("poped element is = %dn", s.stk[s.top]);
        s.top = s.top - 1;
    }
    return(num);
}
/*  Function to display the status of the stack */
void display ()
{
    int i;
    if (s.top == -1)
```

```
    {
        printf ("Stack is empty\n");
        return;
    }
    else
    {
        printf ("\n The status of the stack is \n");
        for (i = s.top; i >= 0; i--)
        {
            printf ("%d\n", s.stk[i]);
        }
    }
    printf ("\n");
}
```

//C Program to Convert Infix to Postfix using Stack

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

char stack[100];
int top = -1;

void push(char x)
{
   stack[++top] = x;
}

char pop()
{
   if(top == -1)
      return -1;
   else
      return stack[top--];
}
int priority(char x)
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
   return 0;
}

int main()
{
   char exp[100];
   char *e, x;
   printf("Enter the expression : ");
   scanf("%s",exp);
   printf("\n");
   e = exp;

   while(*e != '\0')
   {
      if(isalnum(*e))
         printf("%c ",*e);
      else if(*e == '(')
         push(*e);
      else if(*e == ')')
```

```
        {
            while((x = pop()) != '(')
                printf("%c ", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))
                printf("%c ",pop());
            push(*e);
        }
        e++;
    }

    while(top != -1)
    {
        printf("%c ",pop());
    }return 0;
}
```

Output:

Enter the expression: a+b*c

a b c * +

// C Program to Implement a Queue using an Array

```
#include <stdio.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
{
   int choice;
   while (1)
   {
      printf("1.Insert element to queue \n");
      printf("2.Delete element from queue \n");
      printf("3.Display all elements of queue \n");
      printf("4.Quit \n");
      printf("Enter your choice : ");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 4:
         exit(1);
         default:
         printf("Wrong choice \n");
      } /* End of switch */
   } /* End of while */
} /* End of main() */

void insert()
{
   int add_item;
   if (rear == MAX - 1)
   printf("Queue Overflow \n");
   else
```

```
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
} /* End of insert() */

void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
} /* End of delete() */

void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
        printf("\n");
    }
} /* End of display() */
```

// C program to create a linked list and display the elements in the list

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
void main()
{
   struct node
   {
      int num;
      struct node *ptr;
   };
   typedef struct node NODE;
    NODE *head, *first, *temp = 0;
   int count = 0;
   int choice = 1;
   first = 0;
    while (choice)
   {
      head  = (NODE *)malloc(sizeof(NODE));
      printf("Enter the data item\n");
      scanf("%d", &head-> num);
      if (first != 0)
      {
         temp->ptr = head;
         temp = head;
      }
      else
      {
         first = temp = head;
      }
      fflush(stdin);
      printf("Do you want to continue(Type 0 or 1)?\n");
      scanf("%d", &choice);
    }
   temp->ptr = 0;
   /*  reset temp to the beginning */
   temp = first;
   printf("\n status of the linked list is\n");
   while (temp != 0)
   {
      printf("%d=>", temp->num);
      count++;
      temp = temp -> ptr;
   }
   printf("NULL\n");
   printf("No. of nodes in the list = %d\n", count);    }
```

//C Program to perform pre order Tree Traversal

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

/* A binary tree node has data, pointer to left child
 and a pointer to right child */
struct node {
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the
 given data and NULL left and right pointers. */
struct node* newNode(int data) {
    struct node* node = (struct node*) malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

/* Given a binary tree, print its nodes according to the
 "bottom-up" postorder traversal. */
void printPostorder(struct node* node) {
    if (node == NULL)
        return;

    // first recur on left subtree
    printPostorder(node->left);

    // then recur on right subtree
    printPostorder(node->right);

    // now deal with the node
    printf("%d ", node->data);
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node) {
    if (node == NULL)
        return;

    /* first recur on left child */
    printInorder(node->left);
```

```
    /* then print the data of node */
    printf("%d ", node->data);

    /* now recur on right child */
    printInorder(node->right);
}

/* Given a binary tree, print its nodes in inorder*/
void printPreorder(struct node* node) {
    if (node == NULL)
        return;

    /* first print data of node */
    printf("%d ", node->data);

    /* then recur on left sutree */
    printPreorder(node->left);

    /* now recur on right subtree */
    printPreorder(node->right);
}

/* Driver program to test above functions*/
int main() {
    struct node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    printf("\n Preorder traversal of binary tree is \n");
    printPreorder(root);

    printf("\n Inorder traversal of binary tree is \n");
    printInorder(root);

    printf("\n Postorder traversal of binary tree is \n");
    printPostorder(root);

    getchar();
    return 0;
}
```