DSS Programs

Week-1: Introduction to C Lab, Basic Input – Output Operations.

a) Write a C program to print your name and address in line by line.

```c
#include <stdio.h>
int main()
{
    printf("Name: Yo Yo Honey Singh\nAddress: 34-5-12/1/405, Kothapet, Telangana - 500046");
    return 0;
}
```

b) Write a C program to calculate simple interest.

```c
#include <stdio.h>
int main()
{
    float p,t,r;
    printf("Enter principal amount, time and rate: ");
    scanf("%f %f %f",&p,&t,&r);
    float si=p*t*r/100;
    printf("Interest = %f",si);
    return 0;
}
```

c) Write a C program for Swapping of two numbers using a third variable.

```c
#include <stdio.h>
int main()
{
    int a,b,t;
    printf("Enter a and b values: ");
    scanf("%d %d",&a,&b);
    t=a;
    a=b;
    b=t;
    printf("First Swap: a=%d b=%d\n",a,b);
    a=a+b;
    b=a-b;
    a=a-b;
    printf("Second Swap: a=%d b=%d",a,b);
    return 0;
}
```

Week-2: Programs using Control Statements (Conditional).

a) Write a C program to check a character is Vowel or Consonant.

```c
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter an alphabet: ");
    scanf("%c",&ch);
    if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' || ch=='A' || ch=='E' || ch=='I'
|| ch=='O' || ch=='U')
        printf("The alphabet is a vowel.");
    else
        printf("The alphabet is a consonant.");
    return 0;
}
```

b) Write a C program to perform arithmetic operations using switch statement.

```c
#include <stdio.h>
int main()
{
    float num1,num2,result=0;
    int op;
    printf("Choose operation to perform:\n1.Add\n2.Subtract\n3.Multiply\n4.Division\n5.Modulo
Division\n");
    scanf("%d",&op);
    printf("Enter first number: ");
    scanf("%f",&num1);
    printf("Enter second number: ");
    scanf("%f",&num2);
    switch(op)
    {
        case 1:
        result=num1+num2;
        break;
        case 2:
        result=num1-num2;
        break;
        case 3:
        result=num1*num2;
        break;
        case 4:
        result=num1/num2;
        break;
        case 5:
        result=(int)num1%(int)num2;
        break;
        default:
        printf("Invalid operation.\n");
    }
    printf("Result: %f\n",result);
    return 0;
}
```

c) Write a C program find the net salary of employee based on the allowances. (Use else-if ladder).

```c
#include <stdio.h>
int main()
{
    float ts,bs,hra,da,ta,others,pf,it;
    printf("Enter Basic salary: ");
    scanf("%f",&bs);
    printf("Enter House rental allowance: ");
    scanf("%f",&hra);
    printf("Enter Dearness allowance: ");
    scanf("%f",&da);
    printf("Enter Travel allowance: ");
    scanf("%f",&ta);
    printf("Enter Others: ");
    scanf("%f",&others);
    pf = bs*13/100;
    printf("\nProvident Fund: -%f",pf);
    it = bs*15/100;
    printf("\nIncome Tax: -%f",it);
    ts=bs+hra+da+ta+others-(pf+it);
    printf("\nTotal Salary is : %f",ts);
    return 0;
}
```

Week-3: Programs using Control Statements (Iterative).

a) Write a C program to check a given number is palindrome number or not.

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

b) Write a C program to check a number if perfect number or not.

```c
#include <stdio.h>
int main()
{
    int n,i,s=0;
    printf("Enter a number: ");
    scanf("%d",&n);
    for(i=1;i<n;i++)
        if(n%i==0)
            s+=i;
    if(s==n)
        printf("%d is a perfect number.",n);
    else
        printf("%d is not a perfect number.",n);
    return 0;
}
```

c) Write a C program to display Fibonacci numbers up to a given value.

```c
#include <stdio.h>
int main()
{
    int n,i,a=0,b=1,c;
    printf("Enter a number: ");
    scanf("%d",&n);
    printf("Fibonacci Series:\n0 1 ");
    for(i=0;i<n-2;i++)
    {
        c=a+b;
        a=b;
        b=c;
        printf("%d ",c);
    }
    return 0;
}
```

Week-4: Programs using Arrays.

a) Write a C program to find the largest and smallest number among a list of integers.

```c
#include <stdio.h>
int main()
{
    int i,n,max,min;
    printf("Enter the number of array elements: ");
    scanf("%d",&n);
    int a[n];
    printf("Enter the array elements:\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    max=a[0];
    min=a[0];
    for(i=1;i<n;i++)
    {
        if(a[i] > max)
            max=a[i];
        if(a[i] < min)
            min=a[i];
    }
    printf("The largest number is %d\n",max);
    printf("The smallest number is %d",min);
    return 0;
}
```

b) Write a C Program to read an array of n elements and find the sum of all even integers and odd integers.

```c
#include <stdio.h>
int main()
{
    int i,n,esum=0,osum=0;
    printf("Enter the number of array elements: ");
    scanf("%d",&n);
    int a[n];
    printf("Enter the array elements:\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=0;i<n;i++)
    {
        if(a[i]%2==0)
            esum+=a[i];
        else
            osum+=a[i];
    }
    printf("The even number sum is %d\n",esum);
    printf("The odd number sum is %d",osum);
    return 0;
}
```

c) Write a C program to find Multiplication of two matrices.

```c
#include <stdio.h>
int main()
{
    int ra,ca,rb,cb,i,j,k;
    printf("Enter number of rows and columns of matrix A: ");
    scanf("%d %d", &ra,&ca);
    int a[ra][ca];
    printf("Enter elements of Matrix A:\n");
    for (i=0;i<ra;i++)
        for (j=0;j<ca;j++)
            scanf("%d",&a[i][j]);
    printf("Enter number of rows and columns of matrix B: ");
    scanf("%d %d", &rb, &cb);
    if (ca != rb)
        printf("Matrix multiplication is not possible.\n");
    else
    {
        int b[rb][cb],c[ra][cb];
        printf("Enter elements of Matrix-B\n");
        for (i = 0; i < rb; i++)
            for (j = 0; j < cb; j++)
                scanf("%d", &b[i][j]);
        for(int i =0 ; i<ra ;i++)
        {
            for(int j=0;j<cb;j++)
            {
                c[i][j]=0;
                for(int k=0;k<rb;k++)
                    c[i][j] = c[i][j] + a[i][k]*b[k][j];
            }
        }
        printf("\n Product of the matrices:\n");
        for (i = 0; i < ra; i++)
        {
            for (j = 0; j < cb; j++)
                printf("%d\t", c[i][j]);
            printf("\n");
        }
    }
}
```

Week-5: Programs using Strings.

a) Write a C program to demonstrate the string handling functions.

```c
#include<stdio.h>
#include<string.h>
void main()
{
    char s1[30],s2[30];
    int len;
    printf("// Length\n");
    printf("Enter a String: ");
    scanf("%s",s1);
    len=strlen(s1);
    printf("The length of string is %d\n",len);

    printf("\n// Concatenation\n");
    printf("Enter two strings: ");
    scanf("%s %s",s1,s2);
    strcat(s1,s2);
    printf("The concatenated string is %s\n",s1);

    printf("\n// Copying\n");
    printf("Enter a string: ");
    scanf("%s",s1);
    strcpy(s2,s1);
    printf("The copied string is %s\n",s2);

    printf("\n// Comparision\n");
    int x;
    printf("Enter two strings: ");
    scanf("%s %s",s1,s2);
    x=strcmp(s1,s2);
    if(x>0)
        printf("s1 is greater than s2\n");
    else if(x<0)
        printf("s1 is smaller than s2\n");
    else
        printf("The two strings are equal\n");

    printf("\n// Reverse\n");
    printf("Enter a string: ");
    scanf("%s",s1);
    printf("The reverse string is %s\n",strrev(s1));

    printf("\n// Upper & Lower\n");
    printf("Enter a string: ");
    scanf("%s",s1);
    printf("The upper case string is %s\n",strupr(s1));
    printf("The lower case string is %s\n",strlwr(s1));
}
```

b) Write a C program to Check whether a given string is palindrome or not without using string functions.

```c
#include<stdio.h>
void main()
{
    char s[30];
    int i,len=0,count=1;
    printf("Enter a string ");
    scanf("%s",s);
    while(s[len]!='\0')
        len++;
    for(i=0; i < len/2 ; i++)
    {
        if(s[i]!=s[len-1-i])
        {
            count=0;
            break;
        }
    }
    if(count==1)
        printf("%s is Palindrome",s);
    else
        printf("%s is not a Palindrome",s);
}
```

c) Write a C Program to concatenate two string and copy into a new string without using string functions.

```c
#include <stdio.h>
void main()
{
    char s1[100] = "Hello ", s2[100] = "Good Morning.",s3[100];
    int length,i,j;
    length = 0;
    while(s1[length] != '\0')
        length++;
    for (j=0;s2[j]!='\0';j++,length++)
        s1[length] = s2[j];
    s1[length]='\0';
    i=0;
    while(s1[i] != '\0')
    {
        s3[i]=s1[i];
        i++;
    }
    s3[i]='\0';
    printf("After concatenation : %s",s3);
}
```

d) Write a C Program to read two strings and display them in dictionary order.

```c
#include <stdio.h>
#include <string.h>
void main()
{
    char a[100],b[100];
    printf("Enter two strings: ");
    scanf("%s %s",a,b);
    if(strcmp(a,b)<0)
        printf("Dictionary Order:\n%s\n%s",a,b);
    else
        printf("Dictionary Order:\n%s\n%s",b,a);
}
```

<u>**Week-6: Programs using Functions and Structures.**</u>

a) Write a C program to find the factorial of a number using non recursive function.

```c
#include <stdio.h>
void fac(int);
void main()
{
    int n;
    printf("Enter a number: ");
    scanf("%d",&n);
    fac(n);
}
void fac(int n)
{
    int i,f=1;
    for(i=1;i<n+1;i++)
        f*=i;
    printf("Factorial of %d is %d.",n,f);
}
```

b) Write a C program to find the factorial of a number using recursion.

```c
#include <stdio.h>
int fact(int);
void main()
{
    int n,f;
    printf("Enter a number: ");
    scanf("%d",&n);
    f=fact(n);
    printf("The factorial of %d is %d",n,f);
}
int fact(int n)
{
    int f;
    if(n==0)
        return(1);
    else
        return(n*fact(n-1));
}
```

c) Write a C program to find the nth Fibonacci term using recursion.

```c
#include <stdio.h>
int fibo(int);
void main()
{
    int n,f;
    printf("Enter n: ");
    scanf("%d",&n);
    f=fibo(n);
    printf("The nth term of Fibonacci series is %d",f);
}
int fibo(int n)
{
    if(n == 1)
        return 0;
    if(n == 2)
        return 1;
    return (fibo(n-1) + fibo(n-2));
}
```

d) Write a C program to create a student structure containing name, roll number and grade as structure members. Display the name, roll number, and grade of a student.

```c
#include <stdio.h>
struct students
{
    char name[50];
    char rno[20];
    char grade[2];
};
void main()
{
    int i;
    struct students s[3];
    printf("Enter details of the students:\n");
    for(i=0;i<3;i++)
    {
        printf("Enter name of student %d: ",i+1);
        scanf("%s",s[i].name);
        printf("Enter Roll.No of Student %d: ",i+1);
        scanf("%s",s[i].rno);
        printf("Enter Grade of Student %d: ",i+1);
        scanf("%s",s[i].grade);
    }
    for(i=0;i<3;i++)
    {
        printf("\nStudent %d:\n",i+1);
        printf("Name: %s\n",s[i].name);
        printf("Roll.No: %s\n",s[i].rno);
        printf("Grade: %s\n",s[i].grade);
    }
}
```

Week-7: Programs using linear search and binary search.

a) Implement Linear Search

```c
#include <stdio.h>
int main()
{
    int size, num, i, found = 0, pos = -1;
    printf("Enter the number of elements in the array : ");
    scanf("%d", &size);
    int arr[size];
    printf("\nEnter the elements: ");
    for(i=0;i<size;i++)
    scanf("%d", &arr[i]);
    printf("\nEnter the number that has to be searched : ");
    scanf("%d", &num);
    for(i=0;i<size;i++)
    {
        if(arr[i] == num)
        {
            found =1;
            pos=i;
            printf("\nThe number %d is found in the array at position = %d", num,i+1);
            break;
        }
    }
    if (found == 0)
        printf("\n%d does not exist in the array", num);
    return 0;
}
```

b) Implement Binary search

```c
#include <stdio.h>
int main()
{
    int  num, i, n, beg, end, mid, found=0;
    printf("\n Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("\n Enter the elements: ");
    for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
    printf("\n\n Enter the number that has to be searched: ");
    scanf("%d", &num);
    beg = 0, end = n-1;
    while(beg<=end)
    {
        mid = (beg + end)/2;
        if (arr[mid] == num)
        {
            printf("\n %d is present in the array at position %d", num, mid+1);
            found =1;
            break;
        }
        else if (arr[mid]>num)
            end = mid-1;
        else
            beg = mid+1;
    }
    if (beg > end && found == 0)
        printf("\n %d does not exist in the array", num);
    return 0;
}
```

Week-8: Programs using bubble, insertion, and selection sort.

a) Implement Bubble sort

```c
#include <stdio.h>

void bsort(int arr[], int n);

void main()
{
    int i,size;
    printf("Enter number of elements : ");
    scanf("%d",&size);
    int arr[size];
    printf("Enter elements : ");
    for(i=0;i<size;i++)
        scanf("%d",&arr[i]);
    bsort(arr,size);
    printf("Sorted Array:\n");
    for(i=0;i<size;i++)
        printf("%d ",arr[i]);
}

void bsort(int arr[],int n)
{
    int i,j,k;
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-i;j++)
        {
            if(arr[j]>arr[j+1])
            {
                arr[j]+=arr[j+1];
                arr[j+1]=arr[j]-arr[j+1];
                arr[j]-=arr[j+1];
            }
        }
    printf("Pass %d :",i+1);
    for(k=0;k<n;k++)
        printf("%d ",arr[k]);
    printf("\n");
    }
}
```

b) Implement Selection sort

```
#include <stdio.h>

void main()
{
    int n, i, j, pos, temp;
    printf("\nEnter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("\nEnter the elements of the array: ");
    for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
    for(i=0;i<n-1;i++)
    {
        pos = i;
        for(j=i+1;j<n;j++)
            if(arr[j] < arr[pos])
                pos = j;
        temp = arr[i];
        arr[i] = arr[pos];
        arr[pos] = temp;
        printf("Pass %d :",i+1);
        for(j=0;j<n;j++)
            printf("%d ",arr[j]);
        printf("\n");
    }
    printf("\nThe sorted array is:\n");
    for(i=0;i<n;i++)
        printf("%d ", arr[i]);
}
```

c) Implement Insertion sort

```c
#include <stdio.h>

void main()
{
    int i, j, temp, n;
    printf("\n Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("\n Enter the elements of the array: ");
    for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
    for(i=1;i<n;i++)
    {
        temp = arr[i];
        j = i-1;
        while((temp < arr[j]) && (j>=0))
        {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = temp;
        printf("Pass %d :",i+1);
        for(j=0;j<n;j++)
            printf("%d ",arr[j]);
        printf("\n");
    }
    printf("\n The sorted array is: \n");
    for(i=0;i<n;i++)
        printf(" %d\t", arr[i]);
}
```

Week-9: Program to perform Stack operations.

a) Implement stack operations using arrays.

```c
#include <stdio.h>

int top=-1;

void push(int stack[], int val, int n)
{
    if(top != n-1)
        stack[++top] = val;
    else
        printf("\n!!! Stack OVERFLOW\n");
}

void pop(int stack[])
{
    int x;
    if(top != -1)
    {
        x = stack[top--];
        printf("\nThe value deleted is: %d\n", x);
    }
    else
        printf("\n!!! Stack UNDERFLOW\n");
}

void display(int stack[], char c)
{
    int i;
    if(top != -1 && c == 'd')
    {
        for(i = top; i >= 0; i--)
            printf("\n%d", stack[i]);
        printf("\n");
    }
    else if (top != -1 && c == 'p')
        printf("\nThe value at top of stack is: %d\n", stack[top]);
    else
        printf("\n!!! Stack EMPTY.\n");
}

void main()
{
    int n, opt=0, x;
    printf("\nEnter the number of elements in the array: ");
    scanf("%d", &n);
    int stack[n];
    while(opt != 5)
    {
        printf("\n-------------\n\n1. PUSH\n2. POP\n3. PEEK\n4. DISPLAY\n5. EXIT\nEnter Option: ");
        scanf("%d", &opt);
        switch(opt)
        {
        case 1:
            printf("\nEnter the number to push: ");
            scanf("%d", &x);
            push(stack, x, n);
            break;

        case 2:
            pop(stack);
            break;

        case 3:
            display(stack,'p');
            break;

        case 4:
            display(stack,'d');
            break;
        }
    }
}
```

Week-10: Programs to Convert Infix Expression to Postfix, Evaluating Postfix expression.

a) Converting infix expression to postfix expression

```c
#include<stdio.h>
#include<ctype.h>
#define MAX 30

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

void push(char);
char pop();
int priority(char);

int main()
{
    char infix[MAX],postfix[MAX],ch,c;
    int i=0,j=0;
    printf("Enter infix Expression : ");
    scanf("%s",infix);
    push('#');
    while((ch=infix[i++])!='\0')
    {
        if(ch=='(')
            push(ch);
        else if(ch==')')
        {
            while(stack[top]!='(')
                postfix[j++]=pop();
            c=pop();
        }
        else if(isalnum(ch))
            postfix[j++]=ch;
        else
        {
            while(priority(stack[top])>=priority(ch))
                postfix[j++]=pop();
            push(ch);
        }
    }
    while (stack[top] != '#')
        postfix[j++] = pop();
    postfix[j] = '\0';
    printf("\n\nGiven Infix Expn: %s  <=>  Postfix Expn: %s\n", infix, postfix);
}

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

char pop()
{
    return stack[top--];
}

int priority(char op)
{
    switch(op)
    {
        case '#':return 0;
        case '(':return 1;
        case '+':
        case '-':return 2;
        case '*':
        case '/':
        case '%':return 3;
    }
}
```

b) Evaluate the postfix expression

```c
#include <stdio.h>
#include <ctype.h>
#define MAX 100

float st[MAX];
int top=-1;

void push(float st[], float val)
{
    if(top==MAX-1)
        printf("\n STACK OVERFLOW");
    else
    {
        top++;
        st[top]=val;
    }
}

float pop(float st[])
{
    float val=-1;
    if(top==-1)
        printf("\n STACK UNDERFLOW");
    else
    {
        val=st[top];
        top--;
    }
    return val;
}

float evaluatePostfixExp(char exp[])
{
    int i=0;
    float op1, op2, value;
    while(exp[i] != '\0')
    {
        if(isdigit(exp[i]))
            push(st, (float)(exp[i]-'0'));
        else
        {
            op2 = pop(st);
            op1 = pop(st);
            switch(exp[i])
            {
                case '+': value = op1 + op2;
                    break;
                case '-':  value = op1 - op2;
                    break;
                case '/': value = op1 / op2;
                    break;
                case '*':value = op1 * op2;
                    break;
                case '%': value = (int)op1 % (int)op2;
                    break;
            }
            push(st, value);
        }
        i++;
    }
    return(pop(st));
}

void main()
{
    float val;
    char exp[100];
    printf("\n Enter any postfix expression : ");
    gets(exp);
    val = evaluatePostfixExp(exp);
    printf("\n Value of the postfix expression = %.2f", val);
}
```

Week-11: Program to perform Queue Operations.

a) Implement Queue using arrays.

```c
#include <stdio.h>

int front=-1, rear=-1;

void enqueue(int queue[], int val, int n)
{
    if(front == -1 && rear == -1)
    {
        front++;
        queue[++rear] = val;
    }
    else if(rear != n-1)
        queue[++rear] = val;
    else
        printf("\n!!! Queue OVERFLOW\n");
}
void dequeue(int queue[], int n)
{
    int x;
    if(front != -1 && front < n)
    {
        x = queue[front++];
        printf("\nThe value dequeued is: %d\n", x);
    }
    else
        printf("\n!!! Queue UNDERFLOW\n");
}

void display(int queue[], char c)
{
    int i;
    if(rear != -1 && c == 'd')
    {
        for(i = front; i <= rear; i++)
            printf("\n%d", queue[i]);
        printf("\n");
    }
    else if (rear != -1 && c == 'p')
        printf("\nThe value at the front of queue is: %d\n", queue[front]);
    else
        printf("\n!!! Queue EMPTY.\n");
}

void main()
{
    int n, opt=0, x;
    printf("\nEnter the number of elements in the array: ");
    scanf("%d", &n);
    int queue[n];
    while(opt != 5)
    {
        printf("\n------------\n\n1. ENQUEUE\n2. DEQUEUE\n3. PEEK\n4. DISPLAY\n5. EXIT\nEnter Option: ");
        scanf("%d", &opt);
        switch(opt)
        {
        case 1:
            printf("\nEnter the number to insert: ");
            scanf("%d", &x);
            enqueue(queue, x, n);
            break;

        case 2: dequeue(queue, n);
            break;

        case 3: display(queue,'p');
            break;

        case 4: display(queue,'d');
            break;
        }
    }
}
```

Week-12: Programs using singly linked list to perform insert, delete and display.

a) Implement single linked list.

```c
#include <stdio.h>
#include <malloc.h>
struct node
{
    int data;
    struct node *next;
}*head = NULL,*tail = NULL;

void insert_beg()
{
    struct node *new_node;
    int num;
    printf("\nEnter the value : ");
    scanf("%d", &num);
    new_node = (struct node *)malloc(sizeof(struct node));
    new_node -> data = num;
    new_node -> next = head;
    head = tail = new_node;
}

void insert_end()
{
    struct node *new_node;
    int num;
    if(head == NULL)
        printf("\nList is Empty, try inserting at the beginning");
    else
    {
        printf("\nEnter the value : ");
        scanf("%d", &num);
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> data = num;
        new_node -> next = NULL;
        tail -> next = new_node;
    }
}

void insert_middle()
{
    struct node *new_node,*preptr,*ptr;
    int num,pos,i;
    if(head == NULL)
        printf("\nList is Empty, try inserting at the beginning");
    else
    {
        printf("\nEnter the value : ");
        scanf("%d", &num);
        printf("\nEnter the position at which the element has to be inserted : ");
        scanf("%d", &pos);
        ptr = head;
        for(i=1;i<pos;i++)
        {
            preptr = ptr;
            ptr = ptr -> next;
        }
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> data = num;
        new_node -> next = ptr;
        preptr -> next = new_node;
    }
}
```

```c
void delete_beg()
{
    if(head == NULL)
        printf("\nList is Empty, there is nothing to delete");
    else
    {
        int val;
        struct node *ptr;
        ptr = head;
        val = head -> data;
        head = head -> next;
        free(ptr);
        printf("\nDeleted %d from the list",val);
    }
}

void delete_end()
{
    int val;
    if(head == NULL)
        printf("\nList is Empty, there is nothing to delete");
    else if(head == tail)
        delete_beg();
    else
    {
        struct node *ptr;
        val = tail -> data;
        ptr = head;
        while(ptr -> next != tail)
            ptr = ptr -> next;
        ptr -> next = NULL;
        free(tail);
        tail = ptr;
        printf("\nDeleted %d from the list",val);
    }
}

void delete_node()
{
    struct node *ptr, *preptr;
    int i,pos,val;
    if(head == NULL)
        printf("\nList is Empty, there is nothing to delete");
    else
    {
        printf("\nEnter the position of the node which has to be deleted : ");
        scanf("%d", &pos);
        if(pos == 1)
            delete_beg();
        else
        {
            ptr = head -> next;
            for(i=2;i<pos;i++)
            {
                preptr = ptr;
                ptr = ptr -> next;
            }
            preptr -> next = ptr -> next;
            val = ptr -> data;
            free(ptr);
            printf("\nDeleted %d from the list",val);
        }
    }
}
```

```c
void display()
{
    if(head == NULL)
        printf("\nList is Empty, there is nothing to display");
    else
    {
        struct node *ptr;
        ptr = head;
        while(ptr != NULL)
        {
            printf("  %d", ptr -> data);
            ptr = ptr -> next;
        }
    }
}

int main()
{
    int option=0;
    while(option !=8)
    {
        printf("\n\n      ***** MAIN MENU *****\n");
        printf("\n1: Insert a node at the beginning");
        printf("\n2: Insert a node at the end");
        printf("\n3: Insert a node at the given position");
        printf("\n4: Delete a node from the beginning");
        printf("\n5: Delete a node from the end");
        printf("\n6: Delete a node at the given position");
        printf("\n7: Display the list");
        printf("\n8: EXIT");
        printf("\n\nEnter your option : ");
        scanf("%d", &option);
        switch(option)
        {
            case 1: insert_beg();
                break;
            case 2: insert_end();
                break;
            case 3: insert_middle();
                break;
            case 4: delete_beg();
                break;
            case 5: delete_end();
                break;
            case 6: delete_node();
                break;
            case 7: display();
                break;
        }
    }
    return 0;
}
```

Week-13: Programs using singly Linked List to create stack and Queue.

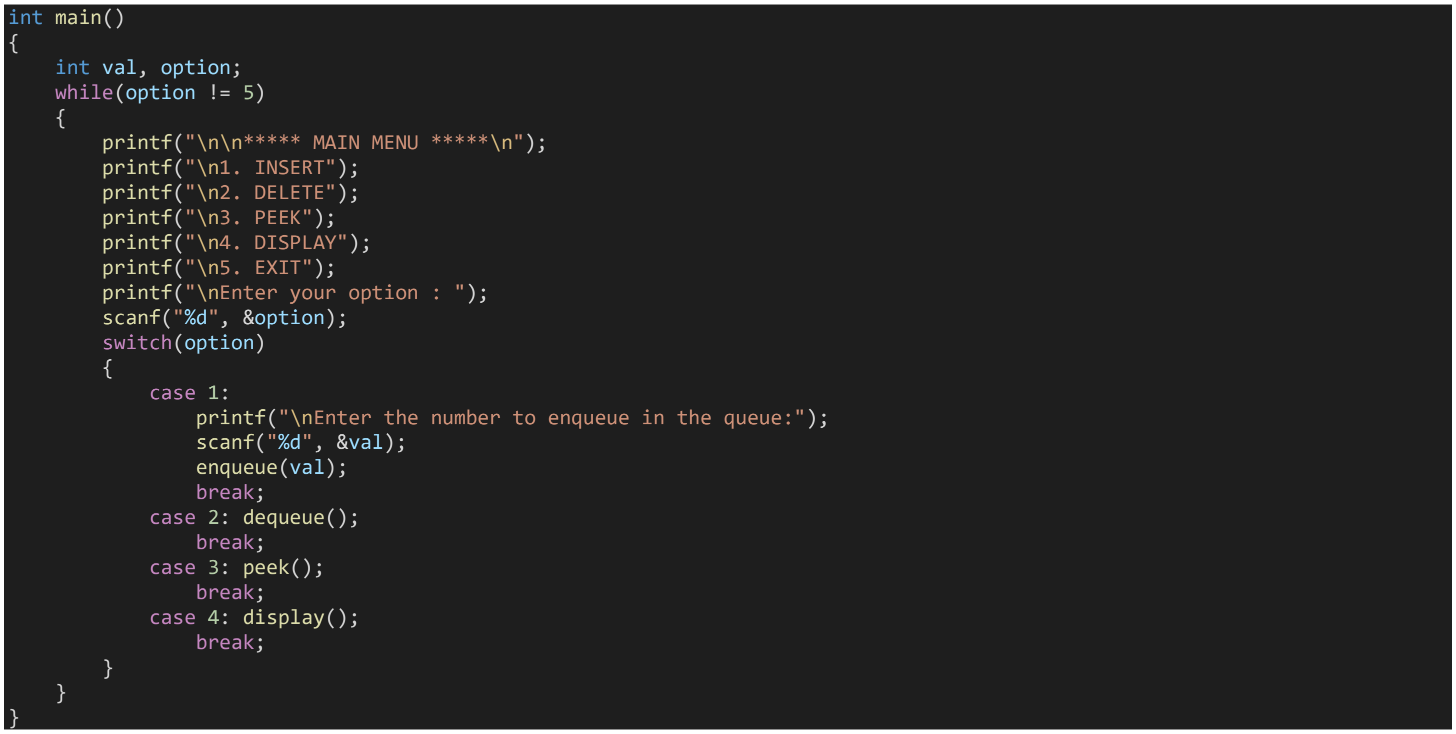
a) Program to implement stack operations using singly linked list

```c
#include <stdio.h>
#include <malloc.h>

struct stack
{
    int data;
    struct stack *next;
}*top = NULL;

void push(int val)
{
    struct stack *ptr;
    ptr = (struct stack*)malloc(sizeof(struct stack));
    ptr -> data = val;
    if(top == NULL)
    {
        ptr -> next = NULL;
        top = ptr;
    }
    else
    {
        ptr -> next = top;
        top = ptr;
    }
}

void display()
{
    struct stack *ptr;
    ptr = top;
    if(top == NULL)
        printf("\nSTACK IS EMPTY");
    else
    {
        while(ptr != NULL)
        {
            printf("\n%d", ptr -> data);
            ptr = ptr -> next;
        }
    }
}

void pop()
{
    struct stack *ptr;
    ptr = top;
    if(top == NULL)
        printf("\nSTACK UNDERFLOW");
    else
    {
        top = top -> next;
        printf("\nThe value being deleted is: %d", ptr -> data);
        free(ptr);
    }
}

void peek()
{
    if(top==NULL)
        printf("\nSTACK IS EMPTY");
    else
        printf("\nThe value at the top of stack is: %d", top -> data);
}
```

```c
void main()
{
    int val, option=0;
    while(option != 5)
    {
        printf("\n\n***** MAIN MENU *****\n");
        printf("\n1. PUSH");
        printf("\n2. POP");
        printf("\n3. PEEK");
        printf("\n4. DISPLAY");
        printf("\n5. EXIT");
        printf("\nEnter your option: ");
        scanf("%d", &option);
        switch(option)
        {
            case 1:
                printf("\nEnter the number to be pushed on stack: ");
                scanf("%d", &val);
                push(val);
                break;
            case 2: pop();
                break;
            case 3: peek();
                break;
            case 4: display();
                break;
        }
    }
}
```

b) Program to implement queue operations using singly linked list.

```c
#include <stdio.h>
#include <malloc.h>

struct queue
{
    int data;
    struct queue *next;
}*front = NULL,*rear = NULL;

void enqueue(int val)
{
    struct queue *ptr;
    ptr = (struct queue*)malloc(sizeof(struct queue));
    ptr -> data = val;
    if(front == NULL)
    {
        front = ptr;
        rear = ptr;
        front -> next = rear -> next = NULL;
    }
    else
    {
        rear -> next = ptr;
        rear = ptr;
        rear -> next = NULL;
    }
}

void dequeue()
{
    struct queue *ptr;
    ptr = front;
    if(front == NULL)
        printf("\nUNDERFLOW");
    else
    {
        front = front -> next;
        printf("\nThe value being deleted is : %d", ptr -> data);
        free(ptr);
    }
}

void display()
{
    struct queue *ptr;
    ptr = front;
    if(ptr == NULL)
        printf("\nQUEUE IS EMPTY");
    else
    {
        printf("\n");
        while(ptr != rear)
        {
            printf("%d  ", ptr -> data);
            ptr = ptr -> next;
        }
        printf("%d  ", ptr -> data);
    }
}

void peek()
{
    if(front == NULL)
        printf("\nQUEUE IS EMPTY");
    else
        printf("\nThe value at front of queue is : %d", front -> data);
}
```



```c
int main()
{
    int val, option;
    while(option != 5)
    {
        printf("\n\n***** MAIN MENU *****\n");
        printf("\n1. INSERT");
        printf("\n2. DELETE");
        printf("\n3. PEEK");
        printf("\n4. DISPLAY");
        printf("\n5. EXIT");
        printf("\nEnter your option : ");
        scanf("%d", &option);
        switch(option)
        {
            case 1:
                printf("\nEnter the number to enqueue in the queue:");
                scanf("%d", &val);
                enqueue(val);
                break;
            case 2: dequeue();
                break;
            case 3: peek();
                break;
            case 4: display();
                break;
        }
    }
}
```

Week-14: Programs using Binary Tree and Tree Traversal.

a) Implement Traversals on Binary Tree using linked list.

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

struct node
{
    struct node *left;
    char data;
    struct node *right;
}*root;

struct node* create()
{
    struct node *newnode;
    char x;
    printf("Enter the data(Type 0 to stop)\n");
    scanf("%c",&x);
    x=getchar();
    if(x=='0')
        return NULL;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=x;
    printf("Enter the left child for %c\n",x);
    newnode->left=create();
    printf("Enter the right child for %c\n",x);
    newnode->right=create();
    return newnode;
}

void insert(struct node *root)
{
    struct node *newnode;
    char ch,x;
    printf("Do you want to enter Left or Right child of %c\n",root->data);
    scanf("%c",&ch);
    ch=getchar();
    if(ch=='l')
    {
        if(root->left==NULL)
        {
            newnode=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data into newnode\n");
            scanf("%c",&x);
            x=getchar();
            newnode->data=x;
            newnode->left=newnode->right=NULL;
            root->left=newnode;
        }
        else
            insert(root->left);
    }
    if(ch=='r')
    {
        if(root->right==NULL)
        {
            newnode=(struct node*)malloc(sizeof(struct node));
            printf("Enter the data into newnode\n");
            scanf("%c",&x);
            x=getchar();
            newnode->data=x;
            newnode->left=newnode->right=NULL;
            root->right=newnode;
        }
        else
            insert(root->right);
    }
}
```

```c
void preorder(struct node *root)
{
    if(root!=NULL)
    {
        printf("%3c",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void inorder(struct node *root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        printf("%3c",root->data);
        inorder(root->right);
    }
}

void postorder(struct node *root)
{
    if(root!=NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%3c",root->data);
    }
}

int main()
{
    int opt=0;
    while(opt != 5)
    {
        printf("\nOperations of Binary Tree...\n");
        printf("1.Create\n2.Preorder\n3.Inorder\n4.Postorder\n5.Exit\n");
        printf("Enter your choice\n");
        scanf("%d",&opt);
        switch(opt)
        {
            case 1:
                printf("Construct the Binary Tree\n");
                root=create();
                break;
            case 2:
                printf("The Preorder Traversal is \n");
                preorder(root);
                break;
            case 3:
                printf("\nThe Inorder Traversal is \n");
                inorder(root);
                break;
            case 4:
                printf("\nThe Postorder Trvarsal is \n");
                postorder(root);
                break;
        }
    }
}
```