**Assignment no.1**

**Program:**

def accept\_set(A,Str): #Accepting the elements and appending to make the set

   n = int(input("Enter the total no. of student who play %s : "%Str))

   for i in range(n) :

      x = input("Enter the name of student %d who play %s : "%((i+1),Str))

      A.append(x)

   print("Set accepted successfully")

def display\_set(A,Str):

   n = len(A)

   if(n == 0) :

      print("\nGroup of Students who play %s =  { }"%Str)

   else :

      print("\nGroup of Students who play %s =  {"%Str,end=' ')

      for i in range(n-1) :

         print("%s,"%A[i],end=' ')

      print("%s }"%A[n-1])

def search\_set(A,X) :

    n = len(A)

    for i in range(n):

       if(A[i] == X) :

          return (1)

    return (0)

def find\_intersection\_set(A,B,C): #Intersection of sets

   for i in range(len(A)):

      flag = search\_set(B,A[i])

      if(flag == 1) :

          C.append(A[i])

def find\_difference\_set(A,B,C): #Difference that is students who play only one game

    for i in range(len(A)):

      flag = search\_set(B,A[i])

      if(flag == 0) :

          C.append(A[i])

def find\_union\_set(A,B,C): #Union of sets

   for i in range(len(A)):

      C.append(A[i])

   for i in range(len(B)):

      flag = search\_set(A,B[i])

      if(flag == 0) :

          C.append(B[i])

def Main() :

   Group\_A = []

   Group\_B = []

   Group\_C = []

   while True :

       print ("\t1 : Accept the Information")

       print ("\t2 : List of students who play both cricket and badminton")

       print ("\t3 : List of students who play either cricket or badminton but not both")

       print ("\t4 : Number of students who play neither cricket nor badminton")

       print ("\t5 : Number of students who play cricket and football but not badminton")

       print ("\t6 : Exit")

       ch = int(input("Enter your choice : "))

       Group\_R = []

       if (ch == 6):

           print ("End of Program")

           break

       elif (ch==1):

           accept\_set(Group\_A,"Cricket")

           accept\_set(Group\_B,"Badminton")

           accept\_set(Group\_C,"Football")

           display\_set(Group\_A,"Cricket")

           display\_set(Group\_B,"Badminton")

           display\_set(Group\_C,"Football")

       elif (ch==2):

           display\_set(Group\_A,"Cricket")

           display\_set(Group\_B,"Badminton")

           find\_intersection\_set(Group\_A,Group\_B,Group\_R)

           display\_set(Group\_R," both Cricket and Badminton")

       elif (ch==3):

           display\_set(Group\_A,"Cricket")

           display\_set(Group\_B,"Badminton")

           R1 = []

           find\_union\_set(Group\_A,Group\_B,R1)

           R2 = []

           find\_intersection\_set(Group\_A,Group\_B,R2)

           find\_difference\_set(R1,R2,Group\_R)

           display\_set(Group\_R," either cricket or badminton but not both")

       elif (ch==4):

           display\_set(Group\_A,"Cricket")

           display\_set(Group\_B,"Badminton")

           display\_set(Group\_C,"Football")

           R1 = []

           find\_union\_set(Group\_A,Group\_B,R1)

           find\_difference\_set(Group\_C,R1,Group\_R)

           display\_set(Group\_R," neither cricket nor badminton")

           print("Number of students who play neither cricket nor badminton = %s"%len(Group\_R))

       elif (ch==5):

           display\_set(Group\_A,"Cricket")

           display\_set(Group\_C,"Football")

           display\_set(Group\_B,"Badminton")

           R1 = []

           find\_intersection\_set(Group\_A,Group\_C,R1)

           find\_difference\_set(R1,Group\_B,Group\_R)

           display\_set(Group\_R,"cricket and football but not badminton")

           print("Number of students who play cricket and football but not badminton = %s"%len(Group\_R))

       else :

           print ("Wrong choice entered !! Try again")

Main()

**Assignment no.2**

**Program:**

def average(l):#Avg Score

    sum = 0

    cnt = 0

    for i in range(len(l)):

        if l[i] != -1:

            sum += l[i]

            cnt += 1

    avg = sum / cnt

    print("Total Marks are : ", sum)

    print("Average Marks are : %.3f"%(avg))

def Maximum(l):#Highest Score

    Max = l[0]

    for i in range(len(l)):

        if l[i] > Max:

            Max = l[i]

    return (Max)

def Minimum(l):#Lowest Score, Use of array and for loop

    for i in range(len(l)):

        if l[i] != -1:

            Min = l[i]

            break

    for j in range(i + 1, len(l)):

        if l[j] != -1 and l[j] < Min:

            Min = l[j]

    return (Min)

def absentCnt(l):#No. of absent students

    cnt = 0

    for i in range(len(l)):

        if l[i] == -1:

            cnt += 1

    return (cnt)

def maxFrequency(l):#Marks with highest frequency

    i = 0

    count = 0

    print(" Marks ----> frequency count ")

    for ele in l:

        if l.index(ele) == i:

            print(ele, "---->", l.count(ele))

            if l.count(ele) > count:

                count = l.count(ele)

                mark = ele

        i += 1

    return (mark, count)

def main():

 print("Enter the no of students and their marks accordingly\n")

 marksInFDS = []

 noStudents = int(input("Enter total number of students : "))

 for i in range(noStudents):

    marks = int(input("Enter marks of Student no. " + str(i + 1) + " : "))

    marksInFDS.append(marks)

 flag = 1

 while flag == 1:

    print("/          MENU        /")

    print("1. The average score of class ")

    print("2. Highest score and lowest score of class ")

    print("3. Count of students who were absent for the test ")

    print("4. Display mark with highest frequency ")

    print("5. Exit ")

    choice = int(input("Enter your choice : "))

    if choice == 1:

        average(marksInFDS)

    elif choice == 2:

        print("Highest score in the class is : ", Maximum(marksInFDS))

        print("Lowest score in the class is : ", Minimum(marksInFDS))

    elif choice == 3:

        print("Count of students who were absent for the test is : ", absentCnt(marksInFDS))

    elif choice == 4:

        mark, count = maxFrequency(marksInFDS)

        print("Highest frequency of marks {0} is {1} ".format(mark, count))

    elif choice == 5:

        print("Program ended")

        flag=0

    else:

        print("Wrong choice")

main()

#END

**Assignment no.3**

**Program:**

def addition\_matrix(M1,M2,M3,r,c) : #Function for addition

    for i in range(r) :

      A = []

      for j in range(c):

        A.append(M1[i][j] + M2[i][j])

      M3.append(A)

def substraction\_matrix(M1,M2,M3,r,c) : #Function for subtraction

   for i in range(r) :

     A = []

     for j in range(c):

       A.append(M1[i][j] - M2[i][j])

     M3.append(A)

def multiplication\_matrix(M1,M2,M3,r1,c1,c2) : #Function for Multiplication

   for i in range(r1) :

      A = []

      for j in range(c2) :

        sum = 0

        for k in range(c1) :

           sum = sum + (M1[i][k] \* M2[k][j])

        A.append(sum)

      M3.append(A)

def find\_transpose\_matrix(M,r,c,T) : #Function for Transpose

   for i in range(c):

     A = []

     for j in range(r):

       A.append(M[j][i])

     T.append(A)

def display(M,r,c):  #Function for displaying matrix

    for i in range(r):

        print("\t\t",end=' ')

        for j in range(c):

            print("%4d"%M[i][j],end=' ')

        print(" ")

def accept\_matrix(M):#Function for accepting matrix

    print("Enter the rows and columns of the matrices : ")

    r=int(input("\t row ="))

    c=int(input("\t column ="))

    print("Enter the elements in matrix : \n")

    for i in range(r):

        A = []

        for j in range(c):

            A.append(int(input()))

        M.append(A)

def options():#Choices

    print("1: Addition")

    print("2: Subtraction\t")

    print("3: Multiplication\t")

    print("4: Transpose of 1st Matrix(M1)\t")

    print("5: Transpose of 2nd Matrix(M2)\t")

    print("6: Exit\t")

def main():

    M1=[]

    M2=[]

    M3=[]

    T1=[]

    accept\_matrix(M1)

    r1=len(M1)

    c1=len(M1[0])

    display(M1,r1,c1)

    accept\_matrix(M2)

    r2=len(M2)

    c2=len(M2[0])

    display(M2,r2,c2)

    while(True):

        options()

        choice=int(input("Enter your choice : "))

        if(choice==1):

         print("Addition of two matrices : \n")

         if(r1==r2 and c1==c2):

           addition\_matrix(M1,M2,M3,r1,c1)

           display(M3,r1,c1)

           M3.clear()

         else:

          print("Addition is not possible")

        elif(choice==2):

          print("Subtraction of two matrices : \n")

          if(r1==r2 and c1==c2):

            substraction\_matrix(M1,M2,M3,r1,c1)

            display(M3,r1,c1)

            M3.clear()

          else:

            print("Subtraction is not possible")

        elif(choice==3):

          print("Multiplication of two matrices :\n")

          if(c1==r2):

            multiplication\_matrix(M1,M2,M3,r1,c1,c2)

            display(M3,r1,c1)

            M3.clear()

          else:

            print("Multiplication not possible(Columns not equal to rows)")

        elif(choice==4):

           print("Transverse of matrix M1 : \n")

           find\_transpose\_matrix(M1,r1,c1,T1)

           display(T1,r1,c1)

           T1.clear()

        elif(choice==5):

           print("Transverse of matrix M2 : \n")

           find\_transpose\_matrix(M2,r2,c2,T1)

           display(T1,r2,c2)

           T1.clear()

        else:

           print("Exiting")

           break

main()

**Assignment no.4**

**Program:**

def accept(A):

    n=int(input("\n Enter total nuber of friends:"))

    print("\n Enter your friends information: ")

    for i in range(n):

        print("\n Enter friend %d information :"%(i+1))

        name=input("\n Enter the name: ")

        mob=input("\n Enter mobile number: ")

        X=[name,mob]

        A.append(X)

    return n

def display(A,n):

    if(n==0):

        print("\n No record found")

    else:

        print("\n Your friends information is : ")

        for i in range(n):

            print("\tFriend no. %d : %s %s"%((i+1),A[i][0],A[i][1]))

def iteractive\_Binary(A,n,s):

    lb=0

    ub=n-1

    while(lb<=ub):

        mid=int((lb+ub)/2)

        if(s==A[mid][0]):

            return mid

        else:

            if(s<A[mid][0]):

                ub=mid-1

            else:

                lb=mid+1

    return -1

def recursive\_Binary(A,lb,ub,s):

    if(lb<=ub):

        mid=int((lb+ub)/2)

        if(s==A[mid][0]):

            return mid

        else:

            if(s<A[mid][0]):

                return recursive\_Binary(A,lb,mid-1,s)

            else:

                return recursive\_Binary(A,mid+1,ub,s)

    else:

      return -1

def Fibonacci(A,n,s):

    f1=0

    f2=1

    f3=f1+f2

    offset=-1

    while(f3<=n):

        f1=f2

        f2=f3

        f3=f1+f2

    while(f3>=0):

        i=min(offset+f1,n-1)

        if(A[i][0]==s):

            return i

        else:

            if(s<A[i][0]):

                f3=f1

                f2=f2-f1

                f1=f3-f2

            else:

                f3=f2

                f2=f1

                f1=f3-f2

                offset=i

    if(f2 and A[n-1] == s):

        return n-1

    return -1

def insert(A,n,s):

    mob=input("\nEnter your friends mobile number : ")

    X=[s,mob]

    A.append(X)

    j=n-1

    while(j>=0):

        if(A[j][0]<=s):

            break

        else:

            A[j+1]=A[j]

        j=j-1

    A[j+1]=X

    display(A,n+1)

def main():

    A=[]

    while True:

        print("\nPress 1.Accept friends information :")

        print("\nPress 2.Iteractive Binary Search :")

        print("\nPress 3.Recursive Bunary Search :")

        print("\nPress 4.Fibonacci Search :")

        print("\nPress 5.Display Record :")

        print("\nPress 6.Exit :")

        ch=int(input("\n Enter your choice :"))

        if(ch==1):

            n=accept(A)

            display(A,n)

        elif(ch==2):

            s=input("\n Enter the friend you want to search: ")

            f=iteractive\_Binary(A,n,s)

            if(f==-1):

                print("\nFriend id not present")

                insert(A,n,s)

                n=n+1

            else:

                print("Friend id present")

        elif(ch==3):

            s=input("\n Enter the friend you want to search: ")

            f=recursive\_Binary(A,0,n-1,s)

            if(f==-1):

                print("\nFriend id not present")

                insert(A,n,s)

                n=n+1

            else:

                print("\nFriend id present")

        elif(ch==4):

            s=input("\n Enter the friend you want to search: ")

            f=Fibonacci(A,n,s)

            if(f==-1):

                print("\nFriend id not present")

                insert(A,n,s)

                n=n+1

            else:

                print("\nFriend id present")

        elif(ch==5):

            display(A,n)

        elif(ch==6):

            break

main()

**Assignment no.5**

**Program:**

def Accept(p):

    p.clear()

    n=int(input("Enter the total number of students : "))

    for i in range(n):

        x=float(input("Enter the percentage of Fe students : "))

        p.append(x)

def Display(p):

    print("FE students percentage :")

    for i in range(len(p)):

        print("%.2f"%p[i])

def Topfive(p):

    print("Top five FE students percentage :")

    for i in range(len(p)-1,len(p)-6,-1):

        print("%.2f"%p[i])

def selection\_sort(p):

    n=len(p)

    for i in range(n-1):

        min\_index=i

        for j in range(i+1,n):

            if(p[j]<p[min\_index]):

                min\_index=j

        temp=p[i]

        p[i]=p[min\_index]

        p[min\_index]=temp

def bubble\_sort(p):

    n=len(p)

    for i in range(1,n):

        for j in range(n-i):

            if(p[j]>p[j+1]):

                temp=p[j]

                p[j]=p[j+1]

                p[j+1]=temp

def main():

    p=[]

    while True:

        print("\n Press 1.Accept students percentage")

        print("\n Press 2.Selection sort")

        print("\n Press 3.Bubble sort")

        print("\n Press 4.Exit")

        ch=int(input("Enter the choice : "))

        if(ch==1):

           Accept(p)

           Display(p)

        elif(ch==2):

           selection\_sort(p)

           if(len(p)>5):

             Topfive(p)

           else:

             Display(p)

        elif(ch==3):

           bubble\_sort(p)

           if(len(p)>5):

            Topfive(p)

           else:

               Display(p)

        else:

            print("End program")

            break

main()

**Assignment no.6**

**Program:**

#Quick Sort

def partition(A, lb, ub):

    pivot = lb

    i = lb + 1

    j = ub

    while i <= j:

        while i <= ub and A[pivot] >= A[i]:

            i = i + 1

        while A[pivot] < A[j]:

            j = j - 1

        if i < j:

            temp = A[i]

            A[i] = A[j]

            A[j] = temp

    temp = A[pivot]

    A[pivot] = A[j]

    A[j] = temp

    return j

def Quick\_Sort(A, lb, ub):

    if lb < ub:

        mid = partition(A, lb, ub)

        Quick\_Sort(A, lb, mid - 1)#recursion

        Quick\_Sort(A, mid + 1, ub)#recursion

def Accept\_Per(A):#Accepting the list

    A.clear()

    n = int(input("Enter the total Strength of class :"))

    for i in range(n):

        x = float(input("Enter the percentage of %d student :" % (i + 1)))

        A.append(x)

def Display\_Per(A):#Displaying percentage of all students

    print("Percentage of students are:")

    for i in range(len(A)):

        print("%.2f" % A[i])

def Display\_Per1(A):#Displaying percentage of top five students

    print("Percentage of top five students are:")

    for i in range(len(A)-1,-1,-1):

        print("%.2f" % A[i])

def main():

    A = []

    while True:

        print("\nPress 1. to accept and display :")

        print("\n Press 2. Quick sort :")

        print("\nPress 3. Exit")

        ch = int(input("Enter your choice:"))

        if ch == 3:

            print("End of program")

            quit()

        elif ch == 1:

            Accept\_Per(A)

            Display\_Per(A)

        elif ch == 2:

            n = len(A)

            Quick\_Sort(A, 0, n - 1)

            if(len(A) > 5) :

             print("Top Five Scores : ")

             for i in range(n-1,n-6,-1) :

               print("\t%.2f"%A[i])

            else:

               Display\_Per1(A)

main()

**Assignment no.7**

**Program:**

/\*ASSIGNMENT 7.

Department of Computer Engineering has student's club named 'Pinnacle Club'.

Students of Second, third and final year of department can be granted membership

on request. Similarly one may cancel the membership of club. First node is reserved

for president of club and last node is reserved for secretary of club. Write C++

program to maintain club memberÃ¢â‚¬Ëœs information using singly linked list. Store

student PRN and Name. Write functions to

a) Add and delete the members as well as president or even secretary.

b) Compute total number of members of club

c) Display members

d) Display list in reverse order using recursion

e) Two linked lists exists for two divisions. Concatenate two lists.

\*/

#include <iostream>

#include <string.h>

using namespace std;

//Node

struct node {

int prn;

string name;

struct node \*next;

};

//Linked List

class list {

node \*head, \*temp;

public:

list() {

head = NULL;

}

node \*create(int val, string n);

void insertEnd();

void insertBeg();

void deleteAt(int i);

void insertAt(int i);

void display();

int count();

void reverse();

void rev(node \*t);

node\* readAt(int i);

void concatenate(list A,list B);

void op();

};

//Create

node\* list::create(int val, string n) {

temp = new(struct node);

if (temp == NULL) {

cout<<"Memory Allocation Failed!"<<endl;

return 0;

} else {

temp -> prn = val;

temp -> name = n;

temp -> next = NULL;

return temp;

}

}

//Insert End

void list::insertEnd() {

int val;

string n;

cout<<"Enter PRN: ";

cin>>val;

cout<<"Enter Name: ";

cin>>n;

struct node \*t = head;

temp = create(val,n);

if (head == NULL) {

head = temp;

head -> next = NULL;

} else {

while ((t -> next) != NULL) {

t = t -> next;

}

temp -> next = NULL;

t -> next = temp;

cout<<"Element Inserted at Last"<<endl;

}

}

//Insert At

void list::insertAt(int i) {

int val,pos = i - 1,counter = 1;

string n;

struct node \*ptr;

struct node \*t = head;

while ((t -> next) != NULL) { //loop to count number of items in linked list.

t = t -> next;

counter++;

}

t = head; //traverse pointer is pointed to head again.

if (i == 1) { //equivalent to insert at start.

insertBeg();

} else if (pos > counter || i <= 0) { //if position is greater than the actual linked list.

cout<<"Entered position is out of scope."<<endl;

} else { //insert at required position.

cout<<"Enter PRN: ";

cin>>val;

cout<<"Enter Name: ";

cin>>n;

temp = create(val,n);

while (pos--) {

ptr = t;

t = t -> next;

}

temp -> next = t;

ptr -> next = temp;

cout<<"Member Inserted at Position: "<<i<<endl;

}

}

//Delete At

void list::deleteAt(int i) {

int val,pos = i - 1,counter = 1;

string n;

struct node \*ptrl,\*ptrr;

struct node \*t = head;

while ((t -> next) != NULL) {

t = t -> next;

counter++;

}

t = head;

if (i == 1) {

ptrl = head;

head = head -> next;

delete ptrl;

} else if (pos > counter || i <= 0) {

cout<<"Entered member doesn't exist."<<endl;

} else {

while (pos--) {

ptrl = t;

t = t -> next;

ptrr = t -> next;

}

ptrl -> next = ptrr;

delete t;

cout<<"Member Deleted at Position: "<<i<<endl;

}

}

//Insert Beg

void list::insertBeg() {

int val;

string n;

cout<<"Enter PRN: ";

cin>>val;

cout<<"Enter Name: ";

cin>>n;

//v = val;

struct node \*t = head;

temp = create(val,n);

if (head == NULL) {

head = temp;

head -> next = NULL;

} else {

temp -> next = head;

head = temp;

cout<<"We have a New President."<<endl;

}

}

//Display

void list::display() {

temp = head;

cout<<"President: ";

cout<< temp -> prn<<" Ã¢â‚¬â€ "<<temp -> name<<" -> ";

if(temp -> next != NULL) {

temp = temp -> next;

}

while (temp -> next != NULL) {

cout<< temp -> prn<<" Ã¢â‚¬â€ "<<temp -> name<<" -> ";

temp = temp -> next;

}

cout<<"Secretary: ";

cout<< temp -> prn<<" Ã¢â‚¬â€ "<<temp -> name<<" -> ";

cout<<"NULL"<<endl;

}

//Count

int list::count() {

temp = head;

int ct = 0;

while (temp != NULL) {

ct++;

temp = temp -> next;

}

return ct;

}

//Concatenate

void list::concatenate(list A,list B) {

struct node \* last,\*last1;

node\* t = A.head;

while (t != NULL) {

int val = t -> prn;

string n = t -> name;

temp = create(val,n);

if (head == NULL) {

head = temp;

head -> next = NULL;

last=head;

} else {

//temp -> next = NULL;

last -> next = t;

last=t;

}

t = t -> next;

}

last -> next = B.head;

t = B.head;

while (t != NULL) {

int val = t -> prn;

string n = t -> name;

temp = create(val,n);

last -> next = temp;

last= temp;

t = t -> next;

}

last->next=NULL;

}

//Accept

void list::op() {

while(1) {

int choice;

cout<<"\nEnter: \n1. Add \n2. Delete \n3. Member's Count \n4. Display \n5. Reverse the List \n0. Prev Menu"<<endl;

cin>>choice;

switch(choice) {

case 1: { //Add

char c;

cout<<"\nEnter: \nA. Add President \nB. Add Secretary \nC. Add Member"<<endl;

cin>>c;

switch(c) {

case 'A':

case 'a':{

insertBeg();

break;

}

case 'B':

case 'b': {

insertEnd();

break;

}

case 'C':

case 'c': {

insertAt(2);

break;

}

}

break;

}

case 2: { //Delete

char c;

cout<<"\nEnter: \nA. Delete President \nB. Delete Secretary \nC. Delete Member"<<endl;

cin>>c;

switch(c) {

case 'A': {

deleteAt(1);

cout<<"Club must have a President. Enter Details"<<endl;

insertBeg();

break;

}

case 'B': {

deleteAt(count());

cout<<"Club must have a Secretary. Enter Details"<<endl;

insertEnd();

break;

}

case 'C': {

int j;

cout<<"Enter Position for Deletion"<<endl;

cin>>j;

deleteAt(j);

break;

}

}

break;

}

case 3: { //Count

cout<<"Count: "<<count()<<endl;

break;

}

case 4: { //Display

if (head == NULL) {

cout<<"NULL"<<endl;

break;

} else {

display();

break;

}

}

case 5: { //Reverse

reverse();

break;

}

case 0: { //Prev Menu

return;

}

}

}

}

//Reverse Recursion

void list::rev(node \*t) {

if(t -> next != NULL) {

rev (t -> next);

}

if(t == head)

cout<<"Secretary: "<<t -> prn<<" Ã¢â‚¬â€ "<<t -> name<<endl;

else if(t -> next == NULL)

cout<<"President: "<<t -> prn<<" Ã¢â‚¬â€ "<<t -> name<<" -> ";

else

cout<<"Member: "<<t -> prn<<" Ã¢â‚¬â€ "<<t -> name<<" -> ";

}

//Reverse

void list::reverse() {

rev(head);

}

//Read At

node\* list::readAt(int i) {

struct node \*t = head;

int c = count();

while(c--) {

t = t-> next;

}

}

//Main

int main() {

list L,X,Y;

int c;

while(1) {

cout<<"Enter: \n1. List A \n2. List B \n3. Concatenate\n0. Exit"<<endl;

cin>>c;

switch(c) {

case 1: cout<<"\nList A:"; X.op(); break;

case 2: cout<<"\nList B:"; Y.op(); break;

case 3: L.concatenate(X,Y); L.display(); break;

case 0: return 0;

}

}

}

**Assignment No.8**

**Program:**

#include<iostream>

using namespace std;

struct node

{ int roll;

struct node \*next;

};

class info

{ node \*head1=NULL,\*temp1=NULL,\*head2=NULL,\*temp2=NULL,\*head=NULL,\*temp=NULL,\*h1=NULL,\*head3=NULL,\*temp3=NULL;

int c,i,f,j,k;

public:

node \*create();

void insert();

void allstud();

void vanila();

void butters();

void uice();

void nice();

void notice();

void ovanila();

void obutters();

void display();

} ;

node \*info::create()

{ node \*p=new(struct node);

cout<<"enter student rollno";

cin>>c;

p->roll=c;

p->next=NULL;

return p;

}

void info::insert()

{

node \*p=create();

if(head==NULL)

{ head=p;

}

else

{ temp=head;

while(temp->next!=NULL)

{ temp=temp->next; }

temp->next=p;

}

}

void info::display()

{ temp=head;

while(temp->next!=NULL)

{ cout<<"\n"<<temp->roll;

temp=temp->next;

} cout<<"\n"<<temp->roll;

}

void info::allstud()

{cout<<"enter no. of students";

cin>>k;

head=NULL;

for(i=0;i<k;i++)

{ insert();

h1=head;

} display();

head=NULL;

}

void info::vanila()

{

cout<<"enter no. of students who like vanila";

cin>>k;

head=NULL;

for(i=0;i<k;i++)

{ insert();

head1=head;

} display();

head=NULL;

}

void info::butters()

{

cout<<"enter no. of students who like butterscotch";

cin>>j;

for(i=0;i<j;i++)

{ insert();

head2=head;

} display();

head=NULL;

}

void info::uice()

{ cout<<"students who like vanila or butterscotch\n";

temp1=head1;

while(temp1!=NULL)

{

node \*p=new(struct node);

p->roll=temp1->roll;

p->next=NULL;

if(head3==NULL)

{ head3=p;

}

else

{ temp3=head3;

while(temp3->next!=NULL)

{ temp3=temp3->next; }

temp3->next=p;

}

temp1=temp1->next;

}

temp2=head2;

while(temp2!=NULL)

{ f=0;

temp1=head1;

while(temp1!=NULL)

{

if(temp2->roll==temp1->roll)

{ f=1; }

temp1=temp1->next;

}

if(f==0)

{

node \*p=new(struct node);

p->roll=temp2->roll;

p->next=NULL;

if(head3==NULL)

{ head3=p;

}

else

{ temp3=head3;

while(temp3->next!=NULL)

{ temp3=temp3->next; }

temp3->next=p;

}

}

temp2=temp2->next;

}

temp3=head3;

while(temp3->next!=NULL)

{ cout<<"\n"<<temp3->roll;

temp3=temp3->next;

} cout<<"\n"<<temp3->roll;

}

void info::ovanila()

{

cout<<"\nstudents like only vanila \n";

temp1=head1;

while(temp1!=NULL)

{ temp2=head2;

f=0;

while(temp2!=NULL)

{ if(temp1->roll==temp2->roll)

{ f=1; }

temp2=temp2->next;

}

if(f==0)

{ cout<<"\n"<<temp1->roll; }

temp1=temp1->next;

}

}

void info::obutters()

{

cout<<"\nstudents like only butterscotch\n";

temp2=head2;

while(temp2!=NULL)

{ temp1=head1;

f=0;

while(temp1!=NULL)

{ if(temp2->roll==temp1->roll)

{ f=1; }

temp1=temp1->next;

}

if(f==0)

{ cout<<"\n"<<temp2->roll; }

temp2=temp2->next;

}

}

void info::nice()

{

cout<<"\nstudents who like both vanila and butterscotch\n";

temp1=head1;

while(temp1!=NULL)

{ temp2=head2;

while(temp2!=NULL)

{ if(temp1->roll==temp2->roll)

{ cout<<"\n"<<temp1->roll; }

temp2=temp2->next;

}

temp1=temp1->next;

}

}

void info::notice()

{

cout<<"\nstudents who like neither vanila nor butterscotch\n";

temp=h1;

while(temp!=NULL)

{ temp3=head3;

f=0;

while(temp3!=NULL)

{ if(temp->roll==temp3->roll)

{ f=1; }

temp3=temp3->next;

}

if(f==0)

{ cout<<"\n"<<temp->roll; }

temp=temp->next;

}

}

int main()

{ info s;

int i;

char ch;

do{

cout<<"\n choice the options";

cout<<"\n 1. To enter all students rollno ";

cout<<"\n 2. To enter the rollno of student who like vanila";

cout<<"\n 3. To enter the rollno of student who like butterscotch";

cout<<"\n 4. To display the rollno of student who like vanila or butterscotch";

cout<<"\n 5. To display the rollno of student who like only vanila";

cout<<"\n 6. To display the rollno of student who like only butterscotch";

cout<<"\n 7. To display the rollno of student who like both vanila and butterscotch ";

cout<<"\n 8. To display the rollno of student who neither like vanila nor butterscotch";

cin>>i;

switch(i)

{ case 1: s.allstud();

break;

case 2: s.vanila();

break;

case 3: s.butters();

break;

case 4: s.uice();

break;

case 5: s.ovanila();

break;

case 6: s. obutters();

break;

case 7: s.nice();

break;

case 8: s.notice();

break;

default: cout<<"\n unknown choice";

}

cout<<"\n do you want to continue enter y/Y \n";

cin>>ch;

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

return 0;

}

**Assignment No.9**

**Program:**

#include<iostream>

#include<string.h>

#define max 50

using namespace std;

class STACK

{

private:

char a[max];

int top;

public:

STACK()

{

top=-1;

}

void push(char);

void reverse();

void convert(char[]);

void palindrome();

};

void STACK::push(char c)

{

top++;

a[top] = c;

a[top+1]='\0';

cout<<endl<<c<<" is pushed on stack ...";

}

void STACK::reverse()

{

char str[max];

cout<<"\n\nReverse string is : ";

for(int i=top,j=0; i>=0; i--,j++)

{

cout<<a[i];

str[j]=a[i];

}

cout<<endl;

}

void STACK::convert(char str[])

{

int j,k,len = strlen(str);

for(j=0, k=0; j<len; j++)

{

if( ( (int)str[j] >= 97 && (int)str[j] <=122 ) || ( (int)str[j] >= 65 && (int)str[j] <=90 ))

{

if( (int)str[j] <=90 )

{

str[k] = (char)( (int)str[j] + 32 );

}else

{

str[k] = str[j];

}

k++;

}

}

str[k]='\0';

cout<<endl<<"Converted String : "<<str<<"\n";

}

void STACK::palindrome()

{

char str[max];

int i,j;

for(i=top,j=0; i>=0; i--,j++)

{

str[j]=a[i];

}

str[j]='\0';

if(strcmp(str,a) == 0)

cout<<"\n\nString is palindrome...";

else

cout<<"\n\nString is not palindrome...";

}

int main()

{

STACK stack;

char str[max];

int i=0;

cout<<"\nEnter string to be reversed and check is it palindrome or not : \n\n";

cin.getline(str , 50);

stack.convert(str);

while(str[i] != '\0')

{

stack.push(str[i]);

i++;

}

stack.palindrome();

stack.reverse();

}

**Assignment No.10**

**Program:**

#include<iostream>

using namespace std;

const int MAX=20;

class Stack

{

char str[MAX];

int top;

public:

Stack()

{

top=-1;

}

void push(char ch);

char pop();

// char getTop();

bool isEmpty();

bool isFull();

void display();

void checkParenthesis();

};

bool Stack::isEmpty()

{

if(top==-1)

return 1;

else return 0;

}

bool Stack::isFull()

{

if(top==MAX-1)

return 1;

else

return 0;

}

void Stack :: display()

{

if(isEmpty()==1)

cout<<"\nStack is empty";

else

{

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

{

cout<<" "<<str[i];

}

}

}

void Stack::push(char ch)

{

if(!isFull())

{

top++;

str[top]=ch;

}

}

char Stack::pop()

{

if(!isEmpty())

{

char ch=str[top];

top--;

return ch;

}

else

{

return '\0';

}

}

void Stack::checkParenthesis()

{

cout<<"\nEnter # as a deliminator after expression(At the end)\n";

cout<<"\nEnter Expression: ";

cin.getline(str,MAX,'#');

char ch;

bool flag=0;

for(int i=0;str[i]!='\0';i++)

{

if(str[i]=='(' || str[i]=='[' || str[i]=='{')

push(str[i]);

if(str[i]==')'||str[i]==']'||str[i]=='}')

{

ch=pop();

if((str[i]==')'&& ch!='(') ||(str[i]==']'&& ch!='[')||(str[i]=='}'&& ch!='{'))

{

cout<<"\nNot parenthesized At "<<i<<" = "<<str[i];

flag=1;

break;

}

}

}

if(isEmpty()==1 && flag==0)

cout<<"\nExpresseion is Well Parenthesized.";

else

cout<<"\nExpression is not Well Parenthesized.";

}

int main()

{

int choice;

do

{

Stack s;

s.checkParenthesis();

cout<<"\nDO you want to continue?{1/0)";

cin>>choice;

}while(choice!=0);

return 0;

}

**Assignment N0.11**

**Program:**

#include <iostream>

#define MAX 10

using namespace std;

struct queue

{ int data[MAX];

int front,rear;

};

class Queue

{ struct queue q;

public:

Queue(){q.front=q.rear=-1;}

int isempty();

int isfull();

void enqueue(int);

int delqueue();

void display();

};

int Queue::isempty()

{

return(q.front==q.rear)?1:0;

}

int Queue::isfull()

{ return(q.rear==MAX-1)?1:0;}

void Queue::enqueue(int x)

{q.data[++q.rear]=x;}

int Queue::delqueue()

{return q.data[++q.front];}

void Queue::display()

{ int i;

cout<<"\n";

for(i=q.front+1;i<=q.rear;i++)

cout<<q.data[i]<<" ";

}

int main()

{ Queue obj;

int ch,x;

do{ cout<<"\n 1. insert job\n 2.delete job\n 3.display\n 4.Exit\n Enter your choice:";

cin>>ch;

switch(ch)

{ case 1: if (!obj.isfull())

{ cout<<"\n Enter data:";

cin>>x;

obj.enqueue(x);

}

else

cout<< "Queue is overflow";

break;

case 2: if(!obj.isempty())

cout<<"\n Deleted Element="<<obj.delqueue();

else

{ cout<<"\n Queue is underflow"; }

cout<<"\nremaining jobs :";

obj.display();

break;

case 3: if (!obj.isempty())

{ cout<<"\n Queue contains:";

obj.display();

}

else

cout<<"\n Queue is empty";

break;

case 4: cout<<"\n Exit";

}

}while(ch!=4);

return 0;

}

**Assignment No.12**

**Program:**

#include<iostream>

//#include

//#include

using namespace std;

#define SIZE 5

// ERROR HANDLINH NOT DOne

// program is not working correct.

//

class dequeue

{

int a[10],front,rear,count;

public:

dequeue();

void add\_at\_beg(int);

void add\_at\_end(int);

void delete\_fr\_front();

void delete\_fr\_rear();

void display();

};

dequeue::dequeue()

{

front=-1;

rear=-1;

count=0;

}

void dequeue::add\_at\_beg(int item)

{

int i;

if(front==-1)

{

front++;

rear++;

a[rear]=item;

count++;

}

else if(rear>=SIZE-1)

{

cout<<"\nInsertion is not possible,overflow!!!!";

}

else

{

for(i=count;i>=0;i--)

{

a[i]=a[i-1];

}

a[i]=item;

count++;

rear++;

}

}

void dequeue::add\_at\_end(int item)

{

if(front==-1)

{

front++;

rear++;

a[rear]=item;

count++;

}

else if(rear>=SIZE-1)

{

cout<<"\nInsertion is not possible,overflow!!!";

return;

}

else

{

a[++rear]=item;

}

}

void dequeue::display()

{

for(int i=front;i<=rear;i++)

{

cout<<a[i]<<" "; }

}

void dequeue::delete\_fr\_front()

{

if(front==-1)

{

cout<<"Deletion is not possible:: Dequeue is empty";

return;

}

else

{

if(front==rear)

{

front=rear=-1;

return;

}

cout<<"The deleted element is "<<a[front];

front=front+1;

}

}

void dequeue::delete\_fr\_rear()

{

if(front==-1)

{

cout<<"Deletion is not possible:Dequeue is empty";

return;

}

else

{

if(front==rear)

{

front=rear=-1;

}

cout<<"The deleted element is "<< a[rear];

rear=rear-1;

}

}

int main()

{

int c,item;

dequeue d1;

do

{

cout<<"\n\n\*\*\*\*DEQUEUE OPERATION\*\*\*\*\n";

cout<<"\n1-Insert at beginning";

cout<<"\n2-Insert at end";

cout<<"\n3\_Display";

cout<<"\n4\_Deletion from front";

cout<<"\n5-Deletion from rear";

cout<<"\n6\_Exit";

cout<<"\nEnter your choice<1-6>:";

cin>>c;

switch(c)

{

case 1:

cout<<"Enter the element to be inserted:";

cin>>item;

d1.add\_at\_beg(item);

break;

case 2:

cout<<"Enter the element to be inserted:";

cin>>item;

d1.add\_at\_end(item);

break;

case 3:

d1.display();

break;

case 4:

d1.delete\_fr\_front();

break;

case 5:

d1.delete\_fr\_rear();

break;

case 6:

exit(1);

break;

default:

cout<<"Invalid choice";

break;

}

}while(c!=7);

return 0;

}

**Assignment No.13**

**Program:**

#include<iostream>

#include<windows.h>

using namespace std;

const int MAX=5;

class PizzaParlour

{

int front,rear;

int orders[MAX];

public:

PizzaParlour()

{

front=rear=-1;

}

bool addOrder(int data);

void serveOrder();

void display();

};

bool PizzaParlour::addOrder(int id){

if(rear==-1)

{

front=rear=0;

orders[rear]=id;

return true;

}

else

{

int pos=(rear+1)%MAX;

if(pos==front)

{

cout<<"\nCafe is Full.Please wait.\n";

return false;

}

else

{

rear=pos;

orders[rear]=id;

return true;

}

}

}

void PizzaParlour::serveOrder()

{

if(front==-1)

{

cout<<"\n No Orders in Cafe.[Cafe is Empty)\n";

return;

}

else

{

cout<<"\n Order No. "<<orders[front]<<" is processed.\n";

if(front==rear) //only one order

{

front=rear=-1;

}

else

{

front=(front+1)%MAX;

}

}

}

void PizzaParlour::display()

{

int i=0;

if(front==-1)

{

cout<<"\nCafe is Empty.No orders.\n";

return;

}

else

{

cout<<"Order Id's: \n";

for(i=front;i!=rear;i=((i+1)%MAX))

{

cout<<orders[i]<<" ";

}

cout<<orders[rear];

}

}

void intro()

{ char name[50]={"\n Cafe \n"};

for(int i=0;name[i]!='\0';i++)

{

Sleep(50);

cout<<name[i];

}

}

int main()

{

int ch,id=0;

PizzaParlour q;

do

{

cout<<"\n-----------------";

intro();

cout<<"-----------------";

cout<<"\n\*\*\*\*Menu\*\*\*\*\*\n";

cout<<"1. Accept order\n";

cout<<"2. Serve order\n";

cout<<"3. Display orders\n";

cout<<"4. Exit";

cout<<"\nChoice: ";

cin>>ch;

switch(ch)

{

case 1:

id++;

if(q.addOrder(id))

{

cout<<"Thank you for order.Order id is : "<<id;

}

else

{

id--;

}

break;

case 2: q.serveOrder();

break;

case 3: q.display();

break;

}

}while(ch!=4);

cout<<"\nThank You.Keep Visiting.";

}