

Write a Python program to store marks scored in subject "Fundamental of Data Structure" by N students in the class. Write functions to compute following:

- a) The average score of class
- b) Highest score and lowest score of class
- c) Count of students who were absent for the test
- d) Display mark with highest frequency

Program:

```
print("Enter the number of students = ")
n=int(input())
print("Subject: Fundamental of data Structure")
b1=[]
s=0
print("Enter the marks of the students: ")
for i in range(n):
    a=int(input())
    b1=b1+[a]
    s=s+a
avg=s/n
print("Average = ", avg , " marks")
count=0
max=b1[0]
min=b1[0]
for i in range(n):
    if i<n-1:
        if b1[i+1]>max:
            max = b1[i+1]
        if b1[i+1]<min:
            min = b1[i+1]
    if b1[i]==0:
        count+=1
print("The number of students who were absent", count)
print("The highest score =", max," and the lowest score =", min)
fr = [None] * n;
v = -1;

for i in range(0,n):
    count = 1
    for j in range(i+1, n):
        if b1[i] == b1[j]:
```

Output:

[illegible]

In second year computer engineering class, group A student's play cricket, group B students play badminton and group C students play football.

Write a Python program using functions to compute following: -

- a) List of students who play both cricket and badminton**
 - b) List of students who play either cricket or badminton but not both**
 - c) Number of students who play neither cricket nor badminton**
 - d) Number of students who play cricket and football but not badminton.**
- (Note- While realizing the group, duplicate entries should be avoided, Do not use SET built-in functions)**

```
A=[]
x = int(input("enter no of students who play cricket : "))
for i in range(0, x):
    A.append(input("Name of students who play cricket : "))
print(A)

B=[]
y = int(input("enter no of students who play badminton : "))
for j in range(0, y):
    B.append(input("Name of students who play badminton : "))
print(B)

C=[]
z = int(input("enter no of students who play football : "))
for k in range(0, z):
    C.append(input("Name of students who play football : "))
print(C)

def AandB():
    AandB = []
    if (len(A) > len(B)):
        for i in range(x):
            for j in range(y):
                if (A[i] == B[j]):
                    AandB.append(A[i])
            print("Students who play both cricket and badminton : ", AandB)
    else:
        for j in range(y):
            for i in range(x):
                if B[j] == A[i]:
                    AandB.append(B[j])
            print("Students who play both cricket and badminton : ", AandB)
    return AandB
```

```

def AorB():
    AorB=[]
    for i in A:
        if i not in B:
            AorB.append(i)
    for j in B:
        if j not in A:
            AorB.append(j)
    print("Students who play either cricket or badminton but not both", AorB)

def onlyFB():
    onlyC=[]
    for k in C:
        if (k not in A) and (k not in B):
            onlyC.append(k)
    if len(onlyC)>0:
        print(len(onlyC), "students play neither cricket nor badminton")
        print("list: ", onlyC)
    elif len(onlyC) == 0:
        print("No student plays only Football, i.e. neither cricket, nor badminton")

def AandCnotB():
    AandCnotB =[]
    AandCnotB=A+C
    AandCnotB=list(dict.fromkeys(AandCnotB))
    for j in B:
        if j in AandCnotB:
            AandCnotB.remove(j)
    print(len(AandCnotB), "students play cricket and football but not badminton")
    print("list: ", AandCnotB)

```

```

AandB()
AorB()
onlyFB()
AandCnotB()

```

OUTPUT :

```

enter no of students who play cricket : 3
Name of students who play cricket : supriya
Name of students who play cricket : sneha
Name of students who play cricket : anu
['supriya', 'sneha', 'anu']
enter no of students who play badminton : 4
Name of students who play badminton : supriya
Name of students who play badminton : sayali
Name of students who play badminton : sarah
Name of students who play badminton : sneha
['supriya', 'sayali', 'sarah', 'sneha']

```

enter no of students who play football : 3

Name of students who play football : supriya

Name of students who play football : swara

Name of students who play football : anu

['supriya', 'swara', 'anu']

Students who play both cricket and badminton :

['supriya', 'sneha']

Students who play either cricket or badminton but not both:

['anu', 'sayali', 'sarah']

1 students play neither cricket nor badminton:

list: ['swara']

2 students play cricket and football but not badminton:

list: ['anu', 'swara

']

Write a python program to compute following computation on matrix:

- a) Addition of two matrices
- b) Subtraction of two matrices
- c) Multiplication of two matrices
- d) Transpose of a matrix

Program:

```
# Program to add two matrices using nested loop
```

```
X = [[1,2,3],
```

```
      [4 ,5,6],
```

```
      [7 ,8,9]]
```

```
Y = [[9,8,7],
```

```
      [6,5,4],
```

```
      [3,2,1]]
```

```
result = [[0,0,0],
```

```
          [0,0,0],
```

```
          [0,0,0]]
```

```
# iterate through rows
```

```
for i in range(len(X)):
```

```
# iterate through columns
```

```
for j in range(len(X[0])):
```

```
    result[i][j] = X[i][j] + Y[i][j]
```

```
for r in result:
```

```
    print(r)
```

```
# program for Subtraction of Two Matrices-
```

```
matrix1 = [[10, 11, 12],
```

```
           [13, 14, 15],
```

```
           [16, 17, 18]]
```

```
matrix2 = [[1, 2, 3],
```

```
           [4, 5, 6],
```

```
           [7, 8, 9]]
```

```
rmatrix = [[0, 0, 0],
```

```
           [0, 0, 0],
```

```
           [0, 0, 0]]
```

```
for i in range(len(matrix1)):
```

```
    for j in range(len(matrix1[0])):
```

```
        rmatrix[i][j] = matrix1[i][j] - matrix2[i][j]
```

```
for r in rmatrix:
```

```
    print(r)
```

```
# program for Multiply Two 3*3 Matrices entered by User
```

```
matOne = []
```

```
print("Enter 9 Elements for First Matrix: ")
```

```
for i in range(3):
```

```
    matOne.append([])
```

```
    for j in range(3):
```

```
        num = int(input())
```

```
        matOne[i].append(num)
```

```
matTwo = []
```

```
print("Enter 9 Elements for Second Matrix: ")
```

```
for i in range(3):
```

```
    matTwo.append([])
```



```

    for j in range(3):
        num = int(input())
        matTwo[i].append(num)

matThree = []
for i in range(3):
    matThree.append([])
    for j in range(3):
        sum = 0
        for k in range(3):
            sum = sum + (matOne[i][k] * matTwo[k][j])
        matThree[i].append(sum)

print("\nMultiplication Result of Two Given Matrix is:")
for i in range(3):
    for j in range(3):
        print(matThree[i][j], end=" ")
    print()

# Program to transpose a matrix using a nested loop

X = [[12,7],
      [4 ,5],
      [3 ,8]]

result = [[0,0,0],
          [0,0,0]]

# iterate through rows
for i in range(len(X)):
    # iterate through columns
    for j in range(len(X[0])):
        result[j][i] = X[i][j]

for r in result:
    print(r)

```

Output:

For addition:

= [[10,10,10],

[10,10,10],

[10,10,10]]

For subtraction:

[[9,9,9],

[9,9,9],

[9,9,9]]

For Multiplication:

Enter 9 elements of first matrix:

1

2

3

4

5

6

7

8

9

Enter 9 elements of second matrix:

1

2

3

4

5

6

7

8

9

The multiplication is:

[[30,66,102],

[36,81,126],

[42,96,150]]

The transpose matrix is:

[12, 4, 3]

[7, 5, 8]

Write a python program to store first year percentage of students in an array.

Write function for sorting array of floating point numbers in ascending order using:

- a) Selection Sort
- b) Bubble Sort and display top five scores

Program:

```
def SelectionSort(arr,n):  
    for i in range(n):  
        Min = i  
        for j in range(i+1,n):  
            if(arr[j]<arr[Min]):  
                Min = j  
        temp=arr[i]  
        arr[i]=arr[Min]  
        arr[Min]=temp
```

```
print(arr)
```

```
def BubbleSort(arr,n):  
    i = 0  
  
    for i in range(n-1):  
        for j in range(0,n-i-1):  
  
            if(arr[j] > arr[j+1]):  
                temp = arr[j]  
                arr[j] = arr[j+1]  
                arr[j+1] = temp  
  
print(arr)  
print("Top Five Scores are...")  
for i in range (len(arr)-1,1,-1):  
    print(arr[i])
```

```

#Driver Code
print("\nHow many students are there?")
n = int(input())
array = []
i=0
for i in range(n):
    print("\n Enter percentage marks")
    item = float(input())
    array.append(item)

print("You have entered following scores...\n")
print(array)
while(True):
    print("Main Menu")
    print("\n 1. Selection Sort")
    print("\n 2. Bubble Sort")
    print("\n 3. Exit")

    if(choice == 1):
        print("\n The sorted Scores are...")
        SelectionSort(array,n)
    elif(choice ==2):
        print("\n The sorted Scores are...")
        BubbleSort(array,n)
    else:
        print("Exiting")
        break

```

Output:

```

How many students are there?
7
Enter percentage marks
78.63
Enter percentage marks
45.88
Enter percentage marks
98.22
Enter percentage marks
94.52
Enter percentage marks
88.71

```

Enter percentage marks

65.32

Enter percentage marks

72.67

You have entered following scores...

[78.63, 45.88, 98.22, 94.52, 88.71, 65.32, 72

Main Menu

1. Selection Sort

2. Bubble Sort

3. Exit

Enter your Choice:

1

The sorted Scores are...

[45.88, 65.32, 72.67, 78.63, 88.71, 94.52,
98.22] Main Menu

1. Selection Sort

2. Bubble Sort

3. Exit

Enter your Choice:

2

The sorted Scores are...

[45.88, 65.32, 72.67, 78.63, 88.71, 94.52,
98.22] Top Five Scores are...

98.22

94.52

88.71

78.63

72.67

Main Menu

1. Selection Sort

2. Bubble Sort

3. Exit

Enter your Choice:

3

Exiting

Write a python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.

Program:

```
def Quick(arr,low,high):
if(low<high):
m=Partition(arr,low,high)
Quick(arr,low,m-1)
Quick(arr,m+1,high)
def Partition(arr,low,high):
pivot = arr[low]
i=low+1
j=high
flag = False
while(not flag):
while(i<=j and arr[i]<=pivot):
i = i + 1
while(i<=j and arr[j]>=pivot):
j = j - 1
if(j<i):
flag = True
else:
temp = arr[i]
arr[i] = arr[j]
```

```

arr[j] = temp
temp = arr[low]
arr[low] = arr[j]
arr[j] = temp
return j
#Driver Code
print("\nHow many students are there?")
n = int(input())
array = []
i=0
for i in range(n):
    print("\n Enter percentage marks")
    item = float(input())
    array.append(item)
print("You have entered following scores...\n")
print(array)
print("\n The sorted Scores are...")
Quick(array,0,n-1)
print(array)
print("Top Five Scores are...")
for i in range (len(array)-1,1,-1):
    print(array[i])

```

Output:

```

How many students are there?
7
Enter percentage marks
78.63
Enter percentage marks
45.88
Enter percentage marks
92.87
Enter percentage marks
98.55
Enter percentage marks
93.58
Enter percentage marks
88.45
Enter percentage marks
71.65
You have entered following scores...
[78.63, 45.88, 92.87, 98.55, 93.58, 88.45, 71.65]

```


The sorted Scores are...

[45.88, 71.65, 78.63, 88.45, 92.87, 93.58, 98.55]

Top Five Scores are...

98.55

93.58

92.87

88.45

78.63

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 member's 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<stdio.h>
#include <iostream>
```

```
#include<string>
using namespace
```

```
std; class list;
```

```
class node
```

```
{
```

```
int prn;
```

```
string name;
```

```
node *next;
```

```
public:
```

```
node(int
```

```
x, string
```

```
nm)
```

```
{
```

```
prn = x; next
```

```
= NULL;
```

```
name = nm;
```

```
}
```

```
friend
```

```
class list;
```

```
};
```

```
class list
```

```
{
```

```
node * start;
```

```
public:
```

```
list()
```

```
{
```

```
start = NULL;
```

```
}
```

```
void
```

```
create();
```

```
void
```

```
display();
```

```
void
```

```
insertAtBeginning();
```

```
void
```

```
insertAtEnd();
```

```
void
```

```
insertAfter();
```

```
void
```

```
deleteAtFirst();
```

```
void
```

```
deleteByValue();
```

```
void
```

```
deleteAtEnd();
```

```
int
```

```
computeTotal();
```

```
void
```

```
sortList();
```

```
void
```

```
concatList(list & q1);
```

```

void
displayRev(node * t);
bool
reverseDisplay() // function is only
for passing start as argument to recursive function
{
if (start == NULL)
return false;
node * temp = start;
displayRev(temp);
// cout << "(President)";
return true;
}
};
void

list::displayRev(node * t)
{
if (t == NULL)
return;
else
{
displayRev(t->next);
cout << "\nPRN NO:" << t->prn << " Name: " << t->name;
}
}
void
list::create()
{
int
no;
string
nam;
if (start == NULL)
{
cout << "Enter PRN number: ";
cin >> no;
cout << "Enter name: ";
cin >> nam;
cout << nam;
start = new
node(no, nam);
cout << "\n===== List Created =====";
}
else
{
cout << "\nList is already created.";
}
}
void
list::display()
{
node * t;
t = start;
if (start == NULL)
cout << "\nList is Empty";
else
{cout << "\n===== List: =====\n";
while (t != NULL){
cout << t->prn << " " << t->name << " \n";
t=t->next;
}
// cout << t->prn << " " << t->name << " \n";
}
}

```

```

}
}
void

list::insertAtBeginning()
{
int
no;
string
nam;
node * temp;
if (start == NULL)
{
create();
}
else
{
cout << "\nEnter PRN Number : ";
cin >> no;
cout << "Enter Name : ";
cin >> nam;
// cout << nam;
temp = new
node(no, nam);
temp->next = start;
start = temp;;
cout << "Inserted " << temp->name << " at
the beginning.";
}
}

void
list::insertAtEnd()
{
int no;
string
nam;
node *
t;
if (start ==
NULL) create();
else
{
cout << "\nEnter PRN Number :
"; cin >> no;
cout << "Enter Name :
"; cin >> nam;
t = start;
while (t->next !=
NULL) t = t->next;
node * p = new
node(no, nam);
t->next = p;
}
}

void
list::insertAfter()
{
int
prev_no;
cout << "\nEnter PRN No. after do you want insert : ";
cin >> prev_no;
node * t;

```

```

t = start;
string
nam;
int
flag = 0, no;
while (t != NULL)
{
if (t->prn == prev_no)
{
flag = 1;
break;
}
t = t->next;
}
if (flag == 1)
{
node * p;
cout << "\nEnter PRN Number : ";
cin >> no;
cout << "Enter Name : ";
cin >> nam;
p=new node(no, nam);
p->next=t->next;
t->next=p;
}
else
{
cout << "\n" << prev_no << " is not in list.";
}
}
void list::
deleteAtFirst()
{
node * t;
if (start == NULL)
cout << "\nClub is Empty..";
else
{
t=start;
start=start->next;
t->next=NULL; // Not necessary

delete t;
cout << "\nPresident deleted..";
}
}
void list::
deleteByValue()
{
int
no, flag = 0;
node * t, *prev;
if (start == NULL)
cout << "\nList/Club is empty";
else
{
cout << "\nEnter PRN No. of member to be deleted : ";
cin >> no;
t=start->next; // t=start if we have to delete president also..start->next is first member
while (t->next != NULL)
{
if (t->prn == no)
{

```

```

flag = 1;
break;
}
prev = t;
t = t->next;
}
if (flag == 1)
{
prev->next=t->next;
t->next=NULL;
delete t;
cout << "\nMember with PRN No: " << no << " is deleted.";
}
else
cout << "\nMember not found in List./President or Secretary cannot be deleted.";
}
}
void list::
deleteAtEnd()
{
node * t, *prev;
t = start;
if (start == NULL)
cout << "\nClub is Empty..";
else
{

while (t->next != NULL)
{
prev = t;
t = t->next;
}
prev->next = NULL;
delete
t;
cout << "\nSecretary Deleted.";
}
}
int
list::computeTotal()
{
node * t;
int
count = 0;
t = start;
if (start == NULL)
{
cout << "\nList is empty.";
return 0;
}
while (t != NULL)
{
count ++;
t = t->next;
}
return count;
}
void
list::sortList()
{
node * i, *j, *last = NULL;
int
tprn;

```

```

string
tname;
if (start == NULL)
{
cout << "\nList is empty.";
return;
}
for (i=start;i->next != NULL;i=i->next)
{
for (j=start;j->next != last;j=j->next)
{

if ((j->prn) > (j->next->prn))

{
tprn = j->prn;
tname = j->name;
j->prn = j->next->prn;
j->name = j->next->name;
j->next->prn = tprn;
j->next->name = tname;
}
}
}
cout << "\n List is sorted.";

display();

}
void
list::concatList(list & q1)
{

node * t, *p;
t = q1.start;
if (t == NULL)
{

cout << "\nList 2 is empty";

return;
}
p = start; // first
list
while (p->next != NULL)
{
p = p->next;
}

p->next = t;
q1.start = NULL; // second
list is set
to
null
cout << "\nAfter concatenation list : \n";

display();
}
int
main()
{
list * l;
int

```



```

choice, selectList;
list
l1, l2;
l = & l1;
X: cout << "\nSelect List\n1.List 1\n2.List 2\nEnter choice : ";

cin >> selectList;
if (selectList == 1)
{
l = & l1;
}
else if (selectList == 2)
{
l = & l2;
}
else
{

cout << "\nWrong list Number.";
goto
X;
}
do
{

cout << "\n1. Create\n2. Insert President\n3. Insert secretary\n4. Insert after position(member)\n";
cout<<"5. Display list\n6. Delete President\n7.Delete Secretary\n8. Delete Member\n9. Find total No. of members\n10.
Sort list\n11. Reselect List";
cout<< "\n12. Combine lists\n13.Reverse Display\n0. Exit\nEnter your choice :\t";
cin >> choice;
switch(choice)

{

case
1: l->create();
break;
case
2: l->insertAtBeginning();

break;
case
3: l->insertAtEnd();

break;
case
4: l->insertAfter();
break;
case
5: l->display();

break;
case
6: l->deleteAtFirst();

break;
case
7: l->deleteAtEnd();

break;

```

case

8: l->deleteByValue();

break;

case

9: cout << "\nTotal members(including President & Secretary) : " << l->computeTotal(); break;

case

10: l->sortList();

break;

case

11:

goto

X;

break;

case

12:

l1.concatList(l2);

break;

case

13:

l->reverseDisplay();

break;

default:

cout << "Wrong choice";

}

}while (choice != 0);

cout << "\n===== GOOD BYE =====\n";

return 0;

}

Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream. Write C++ program to store two sets using linked list. compute and display-

- a) Set of students who like both vanilla and butterscotch
- b) Set of students who like either vanilla or butterscotch or not both
- c) Number of students who like neither vanilla nor butterscotch

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=NUL
L,*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 vanilla";
    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 vanilla 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 vanilla 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 vanilla";
        cout<<"\n  3. To enter the rollno of student who like butterscotch";
        cout<<"\n  4. To display the rollno of student who like vanilla or butterscotch";
        cout<<"\n  5. To display the rollno of student who like only vanilla";
        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 vanilla and butterscotch ";
        cout<<"\n  8. To display the rollno of student who neither like vanilla 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;

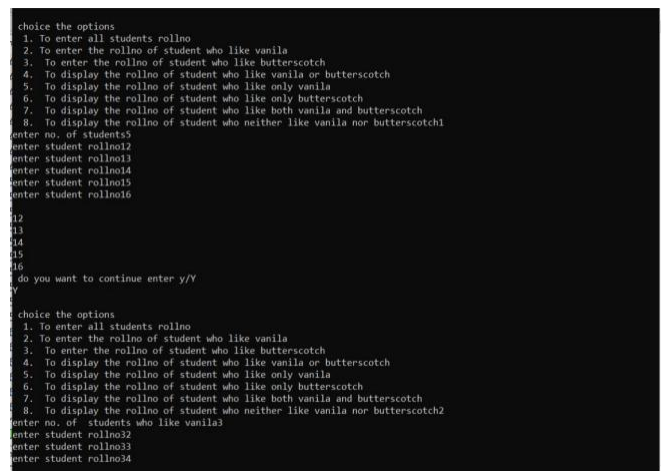
        default: cout<<"\n unknown choice";
    }
    cout<<"\n do you want to continue enter y/Y \n";
    cin>>ch;

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

return 0;
}

```

Output:



```

choice the options
1. To enter all students rollno
2. To enter the rollno of student who like vanilla
3. To enter the rollno of student who like butterscotch
4. To display the rollno of student who like vanilla or butterscotch
5. To display the rollno of student who like only vanilla
6. To display the rollno of student who like only butterscotch
7. To display the rollno of student who like both vanilla and butterscotch
8. To display the rollno of student who neither like vanilla nor butterscotch1
enter no. of students5
enter student rollno2
enter student rollno1
enter student rollno4
enter student rollno5
enter student rollno6
12
13
14
15
16
do you want to continue enter y/Y
Y
choice the options
1. To enter all students rollno
2. To enter the rollno of student who like vanilla
3. To enter the rollno of student who like butterscotch
4. To display the rollno of student who like vanilla or butterscotch
5. To display the rollno of student who like only vanilla
6. To display the rollno of student who like only butterscotch
7. To display the rollno of student who like both vanilla and butterscotch
8. To display the rollno of student who neither like vanilla nor butterscotch2
enter no. of students who like vanilla3
enter student rollno2
enter student rollno3
enter student rollno4

```

```
32
33
34
do you want to continue enter y/Y
Y
choice the options
1. To enter all students rollno
2. To enter the rollno of student who like vanilla
3. To enter the rollno of student who like butterscotch
4. To display the rollno of student who like vanilla or butterscotch
5. To display the rollno of student who like only vanilla
6. To display the rollno of student who like only butterscotch
7. To display the rollno of student who like both vanilla and butterscotch
8. To display the rollno of student who neither like vanilla nor butterscotch2
enter no. of students who like vanilla1
enter student rollno2
2
do you want to continue enter y/Y
Y
choice the options
1. To enter all students rollno
2. To enter the rollno of student who like vanilla
3. To enter the rollno of student who like butterscotch
4. To display the rollno of student who like vanilla or butterscotch
5. To display the rollno of student who like only vanilla
6. To display the rollno of student who like only butterscotch
7. To display the rollno of student who like both vanilla and butterscotch
8. To display the rollno of student who neither like vanilla nor butterscotch3
enter no. of students who like butterscotch2
enter student rollno11
enter student rollno10
11
10
do you want to continue enter y/Y
```

A palindrome is a string of character that's the same forward and backward. Typically, punctuation, capitalization, and spaces are ignored. For example, "Poor Dan is in a droop" is a palindrome, as can be seen by examining the characters "poor danisina droop" and observing that they are the same forward and backward. One way to check for a palindrome is to reverse

the characters in the string and then compare with them the original-in a palindrome, the sequence will be identical. Write C++ program

with functions-

a) To print original string followed by reversed string using stack

b) To check whether given string is palindrome or

not #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();  
  
}
```


Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions

- i. Operands and operator, both must be single character.
- ii. Input Postfix expression must be in a desired format.
- iii. Only '+', '-', '*' and '/' operators are expected.

Program:

```
#include<iostream>
#include<stdlib.h>
#include<cstring>
using namespace std;
int k = 0;//store the length of postfix expression
class stack// declare stack as ADT
{
char st[20];
int top;
public:
stack()
{
top=-1;
}
int empty()//check is stack empty or not
{
if(top== -1)
return 1;
return 0;
}
char gettop()//function will return top element of stack
{
return st[top];
}
void push(char ch)//insert element into stack
{
st[++top]=ch;
}
char pop()//removeing top value from stack
```

```

{ if(!empty())
return st[top--];
}
};
// get priority of operators as per precedence
// higher priority given to operators with higher precedence
// for non operators, return 0
int getpriority(char ch)
{ switch (ch) {
case '/':
case '*': return 2;
case '+':
case '-': return 1;
default : return 0;
}
}
// convert infix expression to postfix using a stack
void infix2postfix(char infix[], char postfix[], int size)
{
stack s;
int priority;
int i = 0;
int k = 0;
char ch;
while (i < size)//read while infix expression not ended
{
ch = infix[i];
if (ch == '(') { // if left paranthesis occur---simply push the
opening parenthesis
s.push(ch);
i++;
continue;
}
if (ch == ')') { // if right parenthesis encountered,
// pop of all the elements and append it to
// the postfix expression till we encounter
// a opening parenthesis
while (!s.empty() && s.gettop() != '(') {
postfix[k++] = s.gettop();
s.pop();
}
// pop off the opening parenthesis
also if (!s.empty()) {
s.pop();
}
}
}

```



```

}
i++;
continue;
}
priority = getpriority(ch);
if (priority == 0) { // we saw an operand
// simply append it to postfix expression
postfix[k++] = ch;
}
else { // we saw an operator
if (s.empty()) { // simply push the operator onto stack
if
// stack is empty
s.push(ch);
}
else {
// pop of all the operators from the stack and
// append it to the postfix expression till we
// see an operator with a lower precedence than
// the current operator
while (!s.empty() && s.gettop() != '(' &&
priority <= getpriority(s.gettop())) {
postfix[k++] = s.gettop();
s.pop();
}
// push the current operator onto stack
s.push(ch);
}
}
i++;
} // pop of the remaining operators present in the stack
// and append it to postfix expression
while (!s.empty()) {
postfix[k++] = s.gettop();
s.pop();
}
postfix[k] = 0; // null terminate the postfix expression
}
int eval(int op1, int op2, char operate)// returns the value when a
specific operator
// operates on two operands
{
switch (operate) {
case '*': return op2 * op1;

```

```

    case '/': return op2 / op1;
    case '+': return op2 + op1;
    case '-': return op2 - op1;
    default : return 0;
}
}
// evaluates the postfix operation
int evalPostfix(char postfix[], int size) {
    stack s;
    int i = 0;
    char ch;
    int val;
    while (i < size) {
        ch = postfix[i];
        if (isdigit(ch)) { // we saw an operand
            // push the digit onto stack
            s.push(ch-'0');
        }
        else { // we saw an operator
            // pop off the top two operands from the
            // stack and evaluate them using the current
            // operator
            int op1 = int(s.gettop());
            s.pop();
            int op2 = int(s.gettop());
            s.pop();
            val = eval(op1, op2, ch); // push the value obtained after
            // evaluating
            // onto the stack
            s.push(val);
        }
        i++;
    }
    return val;
}
// main
int main() {
    char infix[20];
    cout<<"enter the infix expression";
    cin>>infix;
    int size = strlen(infix);
    char postfix[size];
    infix2postfix(infix,postfix,size);
    int val = evalPostfix(postfix, k);
}

```

```
cout<<"\nInfix Expression :: "<<infix;
cout<<"\nPostfix Expression :: "<<postfix;
cout<<"\nExpression evaluates to "<<val;
cout<<endl;
cout<<endl;
return 0;
}
```

Output:

```
student@ubuntu:~$ g++ c2.cpp
student@ubuntu:~$ ./a.out
enter the infix expression(2+3)
closing parenthesis,
Infix Expression :: (2+3)
Postfix Expression :: 23+
Expression evaluates to 5
student@ubuntu:~$ g++ c2.cpp
student@ubuntu:~$ ./a.out
enter the infix expression(7+(4*5/2))-6
closing parenthesis,closing parenthesis,
Infix Expression :: (7+(4*5/2))-6
Postfix Expression :: 745*2/+6-
Expression evaluates to 13*/
```


Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue. Write functions to add job and delete job from queue.

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;
}

```

Output:

```

1. insert job
2.delete job
3.display
4.Exit
Enter your choice:1
Enter data:34

1. insert job

```

2.delete job
3.display
4.Exit
Enter your choice:1
Enter data:64

1. insert job
2.delete job
3.display
4.Exit
Enter your choice:1
Enter data:84

1. insert job
2.delete job
3.display
4.Exit
Enter your choice:1
Enter data:93

1. insert job
2.delete job
3.display
4.Exit
Enter your choice:3
Queue contains:
34 64 84 93

1. insert job
2.delete job
3.display
4.Exit
Enter your choice:2
Deleted Element=34
remaining jobs :
64 84 93

1. insert job
2.delete job
3.display
4.Exit
Enter your choice:3
Queue contains:
64 84 93

1. insert job

2.delete job

3.display

4.Exit

Enter your choice:4

Exit*/

A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a onedimensional array. Write C++ program to simulate deque with functions to add and

delete elements from either end of the deque.

Program:

```
#include<iostream>
using namespace std;
#define SIZE 10
class dequeue {
int a[20],f,r;
public:
dequeue();
void insert_at_beg(int);
void insert_at_end(int);
void delete_fr_front();
void delete_fr_rear();
void show();
};
dequeue::dequeue() {
f=-1;
r=-1;
}
void dequeue::insert_at_end(int i) {
if(r>=SIZE-1) {
cout<<"\n insertion is not possible, overflow!!!!";
} else {
if(f==0) {
f++;
r++;
} else {
r=r+1;
}
a[r]=i;
cout<<"\nInserted item is"<<a[r];
}
}
void dequeue::insert_at_beg(int i) {
if(f==0) {
f=0;
a[++r]=i;
cout<<"\n inserted element is:"<<i;
} else if(f!=0) {
a[--f]=i;
cout<<"\n inserted element is:"<<i;
} else {
cout<<"\n insertion is not possible, overflow!!!!";
}
}
void dequeue::delete_fr_front() {
if(f==0) {
cout<<"deletion is not possible::dequeue is empty";
return;
}
else {
cout<<"the deleted element is:"<<a[f];
```

```

if(f==r) {
f=r=-1;
return;
} else
f=f+1;
}
}
void dequeue::delete_fr_rear() {
if(f==1) {
cout<<"deletion is not possible::dequeue is empty";
return;
}
else {
cout<<"the deleted element is:"<<a[r];
if(f==r) {
f=r=-1;
} else
r=r-1;
}
}
void dequeue::show() {
if(f==1) {
cout<<"Dequeue is empty";
} else {
for(int i=f;i<=r;i++) {
cout<<a[i]<<" ";
}
}
}
int main() {
int c,i;
dequeue d;
Do//perform switch operation {
cout<<"\n 1.insert at beginning";
cout<<"\n 2.insert at end";
cout<<"\n 3.show";
cout<<"\n 4.deletion from front";
cout<<"\n 5.deletion from rear";
cout<<"\n 6.exit";
cout<<"\n enter your choice:";
cin>>c;
switch(c) {
case 1:
cout<<"enter the element to be inserted";
cin>>i;
d.insert_at_beg(i);
break;
case 2:
cout<<"enter the element to be inserted";
cin>>i;
d.insert_at_end(i);
break;
case 3:
d.show();
break;
case 4:

```

```

d.delete_fr_front();
break;
case 5:
d.delete_fr_rear();
break;
case 6:
exit(1);
break;
default:
cout<<"invalid choice";
break;
}
} while(c!=7);
}

```

Output:

```

1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:4
deletion is not possible::dequeue is empty
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:5
deletion is not possible::dequeue is empty
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:1
enter the element to be inserted7
inserted element is:7
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:1
enter the element to be inserted6
insertion is not possible, overflow!!!
1.insert at beginning
2.insert at end
3.show

```

4.deletion from front
5.deletion from rear
6.exit
enter your choice:1
enter the element to be inserted4
insertion is not possible, overflow!!!
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:2
enter the element to be inserted6
Inserted item is6
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:2
enter the element to be inserted4
Inserted item is4
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:3
7 6 4
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:4
the deleted element is:7
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:5
the deleted element is:4
1.insert at beginning
2.insert at end
3.show
4.deletion from front
5.deletion from rear
6.exit
enter your choice:1

enter the element to be inserted:7

inserted element is:7

1.insert at beginning

2.insert at end

3.show

4.deletion from front

5.deletion from rear

6.exit

enter your choice:3

7 6

1.insert at beginning

2.insert at end

3.show

4.deletion from front

5.deletion from rear

6.exit

Enter your choice:6

Pizza parlor accepting maximum M orders.

Orders are served in first come first served basis. Order once placed can not be cancelled.

Write C++ program to simulate the system using circular queue using array.

```
#include<iostream>
#include<cstdlib>
using namespace std;
class pizza
{
    int front,rear,q[5];
public:
    pizza()
    {
        front=-1;
        rear=-1;
    }
    int isfull()
    {
        if((front==0&&rear==4) || front==rear+1)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
    int isempty()
    {
        if(front== -1&&rear== -1)
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
    void add()
    {
        if(isfull()==0)
        {
            cout<<"\n Enter the Pizza ID: ";
            if(front== -1&&rear== -1)
            {
                front=0;
                rear=0;
                cin>>q[rear];
            }
            else
            {
                rear=(rear+1)%5;
                cin>>q[rear];
            }
            char c;
            cout<<" Do you want to add another order ? ";
            cin>>c;
            if(c=='y' || c=='Y')
                add();
        }
        else
        {
            cout<<"\n Orders are full ";
        }
    }
};
```

```

}

}

void serve()
{
if(isempty()==0)
{
if(front==rear)
{
cout<<"\n Order served is : "<<q[front];
front=-1;
rear=-1;
}
else
{
cout<<"\n Order served is : "<<q[front];
front=(front+1)%5;
}
}
else
{
cout<<"\n Orders are empty ";
}
}

void display()
{
if(isempty()==0)
{
for(int
i=front;i!=rear;i=(i+1)%5)
{
cout<<q[i]<<"
<- ";
}
cout<<q[rear];
}
else
{
cout<<"\n Orders are empty";
}
}

void check()
{
int ch;
cout<<"\n\n * * * * PIZZA PARLOUR * * * * \n\n";
cout<<"\n 1. Add a Pizza \n 2. Display the Orders \n 3. Serve a pizza \n 4. Exit \n Enter your choice : ";
cin>>ch;
switch(ch)
{
case 1:
add();
break;

case 2:

display();
break;

case 3:

serve();
break;

```

case 4:

exit(0);

default:

cout<<"Invalid choice ";

check();

}

char ch1;

cout<<"\n Do you want to continue? ";

cin>>ch1;

if(ch1=='y' || ch1=='Y')

check();

}

};

int main()

{

pizza p1;

p1.check();

return 0;

}