```
"""
       Group A - Assignment 1
       In second year computer engineering class, group A students 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)

"""

def accept_set(A,Str):
   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):
   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):
    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):
   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()
quit()
```

```
/*Write C++ program for storing binary number using doubly linked lists.
Write functions a) To compute 1‘s and 2‘s complement b) Add two binary
numbers*/
#include<iostream>
using namespace std;
class binary;
class node
{
      node *prev;
      bool n;
      node*next;
public:
      node()
      {
            prev=next=NULL;
      }
      node(bool b)
      {
            n=b;
            prev=next=NULL;
      }
      friend class binary;
};

class binary
{
      node *start;

      public:
            binary()
            {
                  start=NULL;
            }
            void generateBinary(int no);
            void displayBinary();
            void onesComplement();
            void twoscomplement();
                  binary operator +(binary n1);
      bool addBitAtBegin(bool val)
      {
            node *nodee=new node(val);
            if(start==NULL)
            {
                  start=nodee;
            }
            else
            {
                  nodee->next=start;
                  start->prev=nodee;
                  start=nodee;
            }
            return true;
      }
};

void binary::generateBinary(int no)
{
      bool rem;
```

```
	node *p;
	rem=no%2;
	start=new node(rem);
	no=no/2;
	while(no!=0)
	{
		rem=no%2;
		no=no/2;

	/*
		if(start==NULL)
		{
			start=new node(rem);
		//	cout<<" Start prev: "<<start->prev;
		//	cout<<" Start next: "<<start->next ;

		}
		else
		{
		*/
			p=new node(rem);
			p->next=start;
			start->prev=p;
		//	cout<<" Start prev: "<<start->prev->n;
		//	cout<<"   p->n"<<p->n;
			start=p;

		//}
	}
}
void binary::displayBinary()
{
	node *t;
	t=start;
	while(t!=NULL)
	{
		cout<<t->n;
		t=t->next;
	}

}
void binary::onesComplement()
{
	node *t;
	t=start;

	while(t!=NULL)
	{
		if(t->n==0)
			t->n=1;
		else
			t->n=0;

		t=t->next;

	}
}
binary binary::operator +(binary n1)
```

```
{
	binary sum;
	node *a=start;
	node *b=n1.start;
//	bit *s=sum.start;
	bool carry=false;
	while(a->next!=NULL)
		a=a->next;
	while(b->next!=NULL)
		b=b->next;

	while(a!=NULL && b!=NULL)
	{
		sum.addBitAtBegin((a->n)^(b->n)^carry);
		carry=((a->n&& b->n) || (a->n&& carry) || (b->n && carry));

		a=a->prev;
		b=b->prev;
	}
	while(a!=NULL)
	{
		sum.addBitAtBegin(a->n^carry);
		a=a->prev;
	}
	while(b!=NULL)
	{
		sum.addBitAtBegin(b->n^carry);
		b=b->prev;
	}
	sum.addBitAtBegin(carry);
	return sum;
}
void binary::twoscomplement()
{
	onesComplement();
	bool carry=1;
	node *t;
	t=start;
	while(t->next!=NULL)
	{
		t=t->next;
	}
	while(t!=NULL)
	{
	if(t->n==1&& carry==1)
	{
		t->n=0;
		carry=1;
	}
	else
	if(t->n==0&& carry==1)
	{
		t->n=1;
		carry=0;
	}
	else
	if(carry==0)
	break;
```

```
      t=t->prev;
}
displayBinary();
}
int main()
{
      int num,num1;
      binary n1,n3,n2;
      int choice=1;
      do
      {
            cout<<"\n\n=========Binary Number Operations========\n";
            cout<<"1. Generate binary\n2.One's Complement\n3.Two's
Complement\n4. Addition\n0.Exit\nEnter your choice: ";
            cin>>choice;
            switch(choice)
            {
                  case 1: cout<<"\nENter Number in decimal form: ";
                              cin>>num;
                              n1.generateBinary(num);
                              cout<<"\nBinary Representation: ";
                              n1.displayBinary();
                              break;
                  case 2:cout<<"\nENter Number in decimal form: ";
                              cin>>num;
                              n1.generateBinary(num);
                              cout<<"\nBinary Representation: ";
                              n1.displayBinary();
                              cout<<"\nOnes Complement: ";
                              n1.onesComplement();
                              n1.displayBinary();
                              break;
                  case 3:cout<<"\nENter Number in decimal form: ";
                              cin>>num;
                              n1.generateBinary(num);
                              cout<<"\nBinary Representation: ";
                              n1.displayBinary();
                              cout<<"\nTwos complement; ";
                              n1.twoscomplement();
                              break;
                  case 4: cout<<"\nENter Two Numbers: ";
                              cin>>num>>num1;
                              n1.generateBinary(num);
                              n2.generateBinary(num1);
                              n1.displayBinary();
                              cout<<" + ";
                              n2.displayBinary();
                              cout<<"= ";
                              n3=n1+n2;
                              n3.displayBinary();



            }
      }while(choice!=0);
      n1.generateBinary(7);
```

```
      cout<<"\nBinary Representation: ";
      n1.displayBinary();
//
//    cout<<"\nOnes Complement: ";
//    n1.displayBinary();
      cout<<"\nTwos complement; ";
      n1.twoscomplement();
      return 0;
}
```

```
/*  Assignment No: 9
    In any language program mostly syntax error occurs due to unbalancing
delimiter such as (),{},[]. Write C++ program using stack to check
whether given expression is well parenthesized or not.
   */

#include <iostream>
using namespace std;
#define size 10

class stackexp
{
    int top;
    char stk[size];
public:
    stackexp()
    {
     top=-1;
    }
    void push(char);
    char pop();
    int isfull();
    int isempty();
};

void stackexp::push(char x)
{
    top=top+1;
    stk[top]=x;
}

char stackexp::pop()
{
    char s;
    s=stk[top];
    top=top-1;
    return s;
}

int stackexp::isfull()
{
    if(top==size)
        return 1;
    else
        return 0;
}

int stackexp::isempty()
{
    if(top==-1)
        return 1;
    else
        return 0;
}

int main()
{
    stackexp s1;
```

```
    char exp[20],ch;
    int i=0;
    cout << "\n\t!! Parenthesis Checker..!!!!" << endl; // prints
!!!Hello World!!!
    cout<<"\nEnter the expression to check whether it is in well form or
not :  ";
    cin>>exp;
    if((exp[0]==')')||(exp[0]==']')||(exp[0]=='}'))
    {
        cout<<"\n Invalid Expression.....\n";
        return 0;
    }
    else
    {
        while(exp[i]!='\0')
        {
            ch=exp[i];
            switch(ch)
            {
            case '(':s1.push(ch);break;
            case '[':s1.push(ch);break;
            case '{':s1.push(ch);break;
            case ')':s1.pop();break;
            case ']':s1.pop();break;
            case '}':s1.pop();break;
            }
            i=i+1;
        }
    }
    if(s1.isempty())
    {
        cout<<"\nExpression is well parenthesised...\n";
    }
    else
    {
        cout<<"\nSorry !!! Invalid Expression or not in well
parenthesized....\n";
    }
    return 0;
}
```

```
#include<iostream>
#include<string.h>
using namespace std;
class stackop
{  char  st[20],st1[20];  int top,top1,ss[10],e1,e2,e3,flag;
   public:
    void input();
    void push(char a);
    void pop();
    int pri(char b);
    void eval();
    void push1(int d);
    void pop1();
 };
 int stackop::pri(char b)
 {   if(b=='-')
       return 1;
        if(b=='+')
       return 2;
        if(b=='/')
       return 3;
        if(b=='*')
       return 4;
 }

void stackop::input()
{   char ch[20];   top=-1;   int f=1,l,i=0,j=0;   flag=0;
     cout<<"\n enter the expression\n";
     cin>>ch;
     l=strlen(ch);



     while(i<l)
     {     f=1;
         if(isalpha(ch[i]))
           {    cout<<ch[i]; st1[j]=ch[i]; j++;  flag=1;   }
           if(isdigit(ch[i]))
           {  cout<<ch[i];  st1[j]=ch[i]; j++;   }
         if(ch[i]=='(')
           {    push(ch[i]); }
          if(ch[i]==')')
           {
             while(st[top]!='(')
              {   cout<<st[top]; st1[j]=st[top]; j++; pop();}
                    pop();
           }
           if((ch[i]=='+')||(ch[i]=='-')||(ch[i]=='*')||(ch[i]=='/'))
           {    while(f==1)
                {
                  if(top==-1)
                  {   push(ch[i]); f=0;   }
                  else
                  {   if(st[top]=='(')
                      {   push(ch[i]);  f=0; }
                      else
                      {
                             if((pri(ch[i]))>(pri(st[top])))
```

```
                                {          push(ch[i]); f=0;       }
                                else
                                {      cout<<st[top]; st1[j]=st[top]; j++;
pop();       }

                          }
                        }
                      }
                  }
       i++;
      }
    while(top!=-1)
     { cout<<st[top]; st1[j]=st[top]; j++; pop();   }  cout<<"\n";
     cout<<st1;
}

void stackop::eval()
{  int j=0;  top1=-1;
  if(flag==0)
{
 while(j<strlen(st1))
{
    if(st1[j]=='1')
      push1(1);
if(st1[j]=='2')
      push1(2);
if(st1[j]=='3')
      push1(3);
if(st1[j]=='4')
      push1(4);
if(st1[j]=='5')
      push1(5);
if(st1[j]=='6')
      push1(6);
if(st1[j]=='7')
      push1(7);
if(st1[j]=='8')
      push1(8);
if(st1[j]=='9')
      push1(9);
if(st1[j]=='0')
      push1(0);
if(st1[j]=='+')
{      e1=ss[top1]; pop1();
       e2=ss[top1]; pop1();
       e3=e2+e1;
       push1(e3);
}
if(st1[j]=='-')
 {      e1=ss[top1]; pop1();
       e2=ss[top1]; pop1();
       e3=e2-e1;
       push1(e3);
}

if(st1[j]=='*')
{      e1=ss[top1]; pop1();
       e2=ss[top1]; pop1();
```

```
        e3=e2*e1;
        push1(e3);
}

if(st1[j]=='/')
{       e1=ss[top1]; pop1();
        e2=ss[top1]; pop1();
        e3=e2/e1;
        push1(e3);
} j++;
}
cout<<"\n evaluated value:";
cout<<ss[0];
}
else
{  cout<<"\n cannot evaluate given input";
}
}

void stackop::push(char a)
{    top++; st[top]=a;   }
void stackop::pop()
{  top--;          }
void stackop::push1(int d)
{    top1++; ss[top1]=d;   }
void stackop::pop1()
{  top1--;          }

int main()
{  stackop s;
   s.input();
   s.eval();
   cout<<"\n";
   return 0;
}
```

```
/* Assignment No.11
    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.
    4*/

#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;
}
```

```
/*
 * C++ Program to Implement Priority Queue
 Priority: 1-Low to 10-High
 */
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
using namespace std;

/*
 * Node Declaration
 */
struct node
{
      int priority;
      int info;
      struct node *link;
};
/*
 * Class Priority Queue
 */
class Priority_Queue
{
    private:
        node *front;
    public:
        Priority_Queue()
        {
            front = NULL;
        }
        /*
         * Insert into Priority Queue
         */
        void insert(int item, int priority)
        {
            node *tmp, *q;
            tmp = new node;
            tmp->info = item;
            tmp->priority = priority;
            if (front == NULL || priority < front->priority)
            {
                tmp->link = front;
                front = tmp;
            }
            else
            {
                q = front;
                while (q->link != NULL && q->link->priority <= priority)
                    q=q->link;
                tmp->link = q->link;
                q->link = tmp;
            }
        }
        /*
         * Delete from Priority Queue
         */
        void del()
```

```
        {
            node *tmp;
            if(front == NULL)
                cout<<"Queue Underflow\n";
            else
            {
                tmp = front;
                cout<<"Deleted item is: "<<tmp->info<<endl;
                front = front->link;
                free(tmp);
            }
        }
        /*
         * Print Priority Queue
         */
        void display()
        {
            node *ptr;
            ptr = front;
            if (front == NULL)
                cout<<"Queue is empty\n";
            else
            {    cout<<"Queue is :\n";
                cout<<"Priority       Item\n";
                while(ptr != NULL)
                {
                    cout<<ptr->priority<<"                 "<<ptr-
>info<<endl;
                    ptr = ptr->link;
                }
            }
        }
};
/*
 * Main
 */
int main()
{
    int choice, item, priority;
    Priority_Queue pq;
    do
    {
        cout<<"1.Insert\n";
        cout<<"2.Delete\n";
        cout<<"3.Display\n";
        cout<<"4.Quit\n";
        cout<<"Enter your choice : ";
        cin>>choice;
        switch(choice)
        {
        case 1:
            cout<<"Input the item value to be added in the queue : ";
            cin>>item;
            cout<<"Enter its priority : ";
            cin>>priority;
            pq.insert(item, priority);
            break;
        case 2:
```

```
            pq.del();
            break;
        case 3:
            pq.display();
            break;
        case 4:
            break;
        default :
            cout<<"Wrong choice\n";
        }
    }
    while(choice != 4);
    return 0;
}
```

```
/* Assignment No.=13: 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*/
#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<1-4>:";
            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;

}
```