Q-1 Create a class 'Bank' comprises of data like bank_id, bank_name, brance_name and create another class 'Account' which holds data like ac_no, ac_name, ac_type (saving/current), and balance. Create another class 'Transaction' which holds the functions to perform following operations:

1. Create New Account

2. Deposit & Withdraw (Min 500 balance) in particular

3. List out details of only those accounts whose type is 'current'.

Source Code :

```
 #include <iostream.h>
#include <conio.h>
#include <string.h>
class bank
{
int b_id;
char b_name[10], b_bname[10];
public:
void get_new()
{
 cout<<"ENTER BANK ID:- ";
 cin>>b_id;
 cout<<"ENTER THE BANK NAME:- ";
 cin>>b_name;
 cout<<"ENTER THE BANK BRANCE:- ";
 cin>>b_bname;
}
void display_bank()
{
  cout<<"\nBANK ID:- "<<b_id;
  cout<<"\nBANK NAME:- "<<b_name;
```

```
  cout<<"\nBANK BRANCE:- "<<b_bname;
}
};
class account
{
 int ac_no;
 char ac_name[13];
 public:
 char ac_type;
 float ac_bal;
void get_ac()
{
 cout<<"\nENTER A/C NO:- ";
 cin>>ac_no;
 cout<<"ENTER A/C NAME:- ";
 cin>>ac_name;
 cout<<"ENTER (C-CURRENT OR S-SAVING) A/C:- ";
 cin>>ac_type;
 cout<<"ENTER A/C BALANCE:- ";
 cin>>ac_bal;
}
void display_ac()
{
 cout<<"\nA/C NO:- "<<ac_no;
 cout<<"\nA/C NAME:- "<<ac_name;
 cout<<"\nA/C TYPE:- "<<ac_type;
 cout<<"\nA/C BAL:- "<<ac_bal;
}
};
```

```
class transaction:public bank,public account
{
public:
  float wd;
  int s;
public:
void create()
{
 get_new();
 get_ac();
}
void dep_wit()
{
 cout<<"ENTER 1 FOR DEPOSIT AND 2 FOR WITHDRAW \n";
 cin>>s;
 if(s==1)
 {
 cout<<"ENTER AMOUNT FOR DEPOSIT:- ";
 cin>>wd;
 ac_bal=ac_bal+wd;
 }
 else if(s==2)
 {
  cout<<"ENTER AMOUNT FOR WITHDRAW:- ";
  cin>>wd;
    if(wd>ac_bal)
    {
     cout<<"insufficient bal";
    }
```

```
   else
   {
    ac_bal=ac_bal-wd;
   }
 }
 cout<<"A/C BAL:- "<<ac_bal;
}
void current_details()
{
 if(ac_type=='c' || ac_type=='C')
 {
  cout<<"\n****CURRENT DETAILS****";
  display_bank();
  display_ac();
 }
 else
 {
  cout<<"\nthere is only saving A/C";
 }
 }
};
void main()
{
     clrscr();
     transaction a;
     a.create();
     a.dep_wit();
     a.current_details();
     getch();
```

}

OUTPUT:-

```
ENTER BANK ID:- 2093
ENTER THE BANK NAME:- BOB
ENTER THE BANK BRANCE:- SURAT

ENTER A/C NO:- 098237
ENTER A/C NAME:- ENTER (C-CURRENT OR S-SAVING) A/C:- C
ENTER A/C BALANCE:- 5000
ENTER 1 FOR DEPOSIT AND 2 FOR WITHDRAW
1
ENTER AMOUNT FOR DEPOSIT:- 2000
A/C BAL:- 7000
****CURRENT DETAILS****
BANK ID:- 2093
BANK NAME:- BOB
BANK BRANCE:- SURAT
A/C NO:- 0
A/C NAME:- 98237
A/C TYPE:- C
A/C BAL:- 7000
```

```
ENTER BANK ID:- 2038
ENTER THE BANK NAME:- ICICI
ENTER THE BANK BRANCE:- RAJKOT

ENTER A/C NO:- 86730
ENTER A/C NAME:- ABHAY
ENTER (C-CURRENT OR S-SAVING) A/C:- S
ENTER A/C BALANCE:- 6000
ENTER 1 FOR DEPOSIT AND 2 FOR WITHDRAW
2
ENTER AMOUNT FOR WITHDRAW:- 1500
A/C BAL:- 4500
there is only saving A/C_
```

Q-2 Create a class " word "which stores a string value. Overload +, == for concatenation and comparison operation respectively.

Source Code :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class word
{
char a[20],b[20];
public:
void get()
{
cout<<"enter first string:"<<endl;
cin>>a;
cout<<"enter second string:"<<endl;
cin>>b;
}
void operator +()
{
cout<<"***concatenation***"<<endl;
cout<<a<<" "<<b<<endl;
}
void operator = (word)
{
  cout<<"***comparison***"<<endl;
  if(strcmp(a,b)==0)
  {
```

```
cout<<"equal";
  }
  else
  {
  cout<<"not equal";
  }
}
};
void main()
{
clrscr();
word s;
s.get();
+s;
s=(s);
getch();
}
```

OUTPUT:-

```
enter first string:
adi
enter second string:
adi
***concatenation***
adi adi
***comparison***
equal_
```

```
enter first string:
x066
enter second string:
aditya
***concatenation***
x066 aditya
***comparison***
not equal_
```

Q-3 Create class using multilevel inheritance of student list

1 st class contain roll no and name of student

2 nd class contain marks of three subject

3 rd class contain total and percentage Input data of at least 5 student and display all the information in proper format.

Source Code :

```
#include<iostream.h>
#include<conio.h>
class student
{
 int roll_no;
 char name[15];
 public:
 void get_stud()
 {
 cout<<"Enter rollno:- "<<endl;
 cin>>roll_no;
 cout<<"Enter name:- "<<endl;
 cin>>name;
 }
 void dis_stud()
 {
 cout<<"Roll_no:- "<<roll_no<<endl;
 cout<<"Name:- "<<name<<endl;
 }
};
class marks:public student
```

```
{
public:
  int sub[3];
   void get_marks()
  {
   int i;
    for(i=0;i<3;i++)
            {
              cout<<"\nEnter marks of sub"<<i+1<<":- ";
              cin>>sub[i];
            }
  }
  void dis_marks()
  {
    int j;
    cout<<"\n* * * * subject marks * * * *"<<endl;
       for(j=0;j<3;j++)
       {
            cout<<"sub"<<j+1<<":- "<<sub[j]<<endl;
       }
  }
};
class total:public marks
{
 int total;
 float per;
```

```
 public:
 total()
 {
  total=0;
 }
 void get_total()
 {
 cout<<"\n* * * student detail * * *"<<endl;
   int i;
     for(i=0;i<3;i++)
     {
       total=total+sub[i];
     }
     per=total*100/300;
 }
 void dis_total()
 {
   dis_stud();
   dis_marks();
   cout<<"\n total:- "<<total<<"/300"<<endl;
   cout<<"percentage:- "<<per<<endl;
 }
};
void main()
{
clrscr();
```

```
total s;
s.get_stud();
s.get_marks();
s.get_total();
s.dis_total();
getch();
```

OUTPUT:-

```
Enter rollno:-
101
Enter name:-
Tony

Enter marks of sub1:- 78

Enter marks of sub2:- 88

Enter marks of sub3:- 80

* * * student detail * * *
Roll_no:- 101
Name:- Tony


* * * * subject marks * * * *
sub1:- 78
sub2:- 88
sub3:- 80

total:- 246/300
percentage:- 82
```

Q-4 Create a book class (bookid,bookname,year,publication) & student class (rollno,name,year,books). Display list of books borrowed by student from library.

Source Code :

```
#include<iostream.h>

#include<conio.h>

class book

{

int book_id , t b_year;

char book_name[20],  pub[30];

public:

void get_book()

{

cout<<"Enter book id:- ";

cin>>book_id;

cout<<"Enter book name:- ";

cin>>book_name;

cout<<"Enter year:- ";

cin>>b_year;

cout<<"Enter publication name:- ";

cin>>pub;

}

};

class student:public book

{

int roll_no, year;

char name[15],  s_book[15];
```

```
public:
void get_student()
{
cout<<"****Student detail****"<<endl;
cout<<"Enter roll no:- ";
cin>>roll_no;
cout<<"Enter student nane:- ";
cin>>name;
cout<<"year:- ";
cin>>year;
cout<<"Enter book name to borrowed:- ";
cin>>s_book;
}
void b_book()
{
 cout<<"***Book Borrowed by student***"<<endl;
 cout<<s_book;
}
};
void main()
{
 clrscr();
 student s;
 s.get_book();
 s.get_student();
 s.b_book();
```

```
 getch();
}
```

OUTPUT:-

```
Enter book id:- 208
Enter book name:- c++
Enter year:- 2015
Enter publication name:- naveet
****Student detail****
Enter roll no:- 66
Enter student nane:- aditya
year:- 2019
Enter book name to borrowed:- c++
***Book Borrowed by student***
c++
```

Q-5 Create an event class, create dynamic objects of event class and release the memory of the created object before program terminates.

Source Code:

```
#include<iostream.h>
#include<conio.h>
class event
{
int a[10],n;
public:
event()
{
cout<<"Enter number of value to be entered: ";
cin>>n;
cout<<"constructor is called..."<<endl;
   for(int i=0;i<n;i++)
        {
        cout<<"enter the value of a["<<i+1<<"]:- ";
        cin>>a[i];
        }
}
void dis()
{
for(int i=0;i<n;i++)
  {
        cout<<"a["<<i+1<<"];- "<<a[i]<<endl;
   }
}
```

```
~event()
{
cout<<"destructor is called"<<endl;
}
};
void main()
{
clrscr();
event e;
e.dis();
getch();
}
```

OUTPUT:-

```
Enter number of value to be entered: 3
constructor is called...
enter the value of a[1]:- 1
enter the value of a[2]:- 2
enter the value of a[3]:- 3
a[1]:- 1
a[2]:- 2
a[3]:- 3
destructor is called
Enter number of value to be entered: _
```

Q-6 Create a program to implement stack with its operations

Source Code :

```
#include<iostream.h>
#include<conio.h>
class stack
{
int a[10], top;
public:
 stack()
 {
  top=0;
 }
 void push(int);
 void pop();
 void disp();
};
void stack::push(int l)
{
 if(top==4)
 {
  cout<<"stack is full";
 }
 else
 {
   a[top]=l;
   top++;
```

```
 }
}
void stack::pop()
{
 if(top==0)
 {
  cout<<"stack is empty"<<endl;
 }
 else
 {
   a[top]=0;
   top--;
 }
}
void stack::disp()
{
  int i;
  cout<<"*****stack*****"<<endl;
  for(i=top-1;i>=0;i--)
  {
   cout<<a[i]<<endl;
  }
}
void main()
{
clrscr();
```

```
 stack s;
 s.push(1);
 s.push(2);
 s.push(3);
 s.push(4);
 s.disp();
 s.pop();
 s.push(5);
 s.disp();
 getch();
}
```

OUTPUT:

```
*****stack*****
4
3
2
1
*****stack*****
5
3
2
1
```

Q-7 Create a program to implement infix to postfix conversion.

Source Code :

```
#include<iostream.h>
#include<string>
#define MAX 20
char stk[20];
int top=-1;
void push(char oper)
{
   if(top==MAX-1)
   {
      cout<<"stackfull!!!!";
   }
   else
   {
      top++;
      stk[top]=oper;
   }
}
char pop()
{
   char ch;
   if(top==-1)
   {
      cout<<"stackempty!!!!";
   }
```

```
    else
    {
        ch=stk[top];
        stk[top]='\0';
        top--;
        return(ch);
    }
    return 0;
}
int priority ( char alpha )
{
    if(alpha == '+' || alpha =='-')
    {
        return(1);
    }
    if(alpha == '*' || alpha =='/')
    {
        return(2);
    }
    if(alpha == '$')
    {
        return(3);
    }
    return 0;
}
string convert(string infix)
```

```
{
    int i=0;
    string postfix = "";
    while(infix[i]!='\0')
    {
        if(infix[i]>='a' && infix[i]<='z'|| infix[i]>='A'&& infix[i]<='Z')
        {
            postfix.insert(postfix.end(),infix[i]);
            i++;
        }
        else if(infix[i]=='(' || infix[i]=='{'  || infix[i]=='[')
        {
            push(infix[i]);
            i++;
        }
        else if(infix[i]==')' || infix[i]=='}'  || infix[i]==']')
        {
            if(infix[i]==')')
            {
                while(stk[top]!='(')
                {           postfix.insert(postfix.end(),pop());
                }
                pop();
                i++;
            }
            if(infix[i]==']')
```

```
        {
            while(stk[top]!='[')
            {
                postfix.insert(postfix.end(),pop());
            }
            pop();
            i++;
        }
        if(infix[i]=='}')
        {
            while(stk[top]!='{')
            {
                postfix.insert(postfix.end(),pop());
            }
            pop();
            i++;
        }
    }
    else
    {
        if(top==-1)
        {
            push(infix[i]);
            i++;
        }
        else if( priority(infix[i]) <= priority(stk[top])) {
```

```
            postfix.insert(postfix.end(),pop());
            while(priority(stk[top]) == priority(infix[i])){
                postfix.insert(postfix.end(),pop());
                if(top < 0) {
                    break;
                }
            }
            push(infix[i]);
            i++;
        }
        else if(priority(infix[i]) > priority(stk[top])) {
            push(infix[i]);
            i++;
        }
    }
  }
  while(top!=-1)
  {
    postfix.insert(postfix.end(),pop());
  }
  cout<<"The converted postfix string is : "<<postfix; //it will print postfix
conversion
  return postfix;
}
int main()
{
  int cont;
```

```
    string infix, postfix;
    cout<<"\nEnter the infix expression : "; //enter the expression
    cin>>infix;
    postfix = convert(infix);
    return 0;
}
```

Output:

```
Enter the infix expression : a=b*2+5
The converted postfix string is : ab*=+25
```

Q-8 Create a program to implement simple queue with its operations.

Source Code :

```
#include<iostream.h>
class Queue {
public:
   int front, rear, size;
   unsigned capacity;
   int* array;
};
Queue* createQueue(unsigned capacity)
{
   Queue* queue = new Queue();
   queue->capacity = capacity;
   queue->front = queue->size = 0;
   queue->rear = capacity - 1;
   queue->array = new int[queue->capacity];
   return queue;
}
int isFull(Queue* queue)
{
   return (queue->size == queue->capacity);
}
int isEmpty(Queue* queue)
{
   return (queue->size == 0);
}
```

```
void enqueue(Queue* queue, int item)
{
   if (isFull(queue))
      return;
   queue->rear = (queue->rear + 1)
              % queue->capacity;
   queue->array[queue->rear] = item;
   queue->size = queue->size + 1;
   cout << item << " enqueued to queue\n";
}
int dequeue(Queue* queue)
{
   if (isEmpty(queue))
      return INT_MIN;
   int item = queue->array[queue->front];
   queue->front = (queue->front + 1)
    % queue->capacity;
   queue->size = queue->size - 1;
   return item;
}
int front(Queue* queue)
{
   if (isEmpty(queue))
      return INT_MIN;
   return queue->array[queue->front];
}
```

```
int rear(Queue* queue)
{
   if (isEmpty(queue))
      return INT_MIN;
   return queue->array[queue->rear];
}
int main()
{
   Queue* queue = createQueue(1000);
   enqueue(queue, 10);
   enqueue(queue, 20);
   enqueue(queue, 30);
   enqueue(queue, 40);
   cout << dequeue(queue)
       << " dequeued from queue\n";
   cout << "Front item is "
       << front(queue) << endl;
   cout << "Rear item is "
       << rear(queue) << endl;
  return 0;
}
```

Output:

```
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
10 dequeued from queue
Front item is 20.
```