Binary File

Write a function in C++ to search for a BookNo from a binary file "BOOK.DAT", assuming the binary file is containing the objects of the following class.

```
class BOOK
{
     int Bno;
    char Title[20]; public:
    int RBno( )
   {    return Bno;
    }
    void Enter( )
    { cin>>Bno;gets(Title);
    }
    void Display( )
    {
                        cout<<Bno<<Title<<endl;
    }
};
```

Ans:

```
void BookSearch()
{
fstream FIL; FIL.open("BOOK.DAT",ios::binary|ios::in); BOOK B;
int bn,Found=0;
cout<<"Enter Book Num to search…";
cin>>bn;
while (FIL.read((char*)&S,sizeof(S))) if (B.RBno( )==bn)
{       B.Display( ); Found++;
}
if (Found==0) cout<<"Sorry! Book not found!!!"<<endl;
FIL.close( );
}
```

Write a function in C++ to add new objects at the bottom of a binary file "STUDENT.DAT", assuming the binary file is containing the objects of the following class.

```
class STUD
{   int Rno;
    char Name[20]; public:
    void Enter( )
    { cin>>Rno; gets(Name);
    }
    void Display( )
    { cout<<Rno<<Name<<endl;
    }
};
```

Ans:

```
void Addnew()
{
fstream FIL;
FIL.open("STUDENT.DAT",ios::binary|ios::app);
STUD S;
char CH;
do
{ S.Enter();
FIL.write((char*)&S,sizeof(S));
cout<<"More(Y/N)?";
cin>>CH;
                    }
while(CH!='Y');
 FIL.close();}
```

Write a function in C++ to add new objects at the bottom of a binary file "STUDENT.DAT", assuming the binary file is containing the objects of the following class.

```
class STUD
{ int Rno;
  char Name[20]; public:
  void Enter( )
  { cin>>Rno;gets(Name);
  }
  void Display( )
  { cout<<Rno<<Name<<endl;
}
};
```

Ans

```
void Addnew( )
{fstream FIL;
FIL.open("STUDENT.DAT",ios::binary|ios::app);
STUD S;
char CH; do
{   S.Enter();
FIL.write((char*)&S,sizeof(S));
cout<<"More(Y/N)?";
cin>>CH;
}
while(CH!='Y');
FIL.close( );
}
```

STRUCTURE

Write a function in C++ to perform Delete operation on a dynamically allocated Queue containing Members details as given in the following definition of NODE:

```
struct NODE
{
long Mno //Member Number
char Mname[20]; //Member Name
NODE *Link;
};
```

Ans

```
class Queue
{
NODE *Front, *Rear; public:
Queue ( ) {Front = NULL; Rear = NULL; } void QueAdd ( );
void QueDel ( ); void QueDis ( );
~Queue();
} ;
void Queue: :QueDel ( )
{
if (Front!=NULL)
{
NODE *Temp=Front; cout<<Front->Mno<< " " ;
cout<<Front->Mname<< "Deleted"; Front=Front->Link;
delete Temp;
if (Front==NULL) Rear=NULL;
}
else
cout<<"Underflow ! Queue is empty. .";
}
```

Write a function QUEDEL( ) in C++ to display and delete an element from a dynamically allocated Queue containing nodes of the following given structure:

```
struct NODE
{ int Itemno;
  char Itemname[20];
  NODE *Link;
} ;
```

Ans)

```
class Queue
{
  Node *Front, *Rear; public:
  QUEUE( ) //Constructor to initialize Front and Rear
      { Front = NULL;
        Rear = NULL;
      }
  void QUEINS( );              //Function to insert a node
  void QUEDEL( );            //Function to delete a node void QUEDISP( );      //Function to
  display nodes
   ~Queue();               //Destructor to delete all nodes
};
void Queue::QUEDEL( )
{ if (Front!=NULL)
  {NODE *Temp=Front; cout<<Front->Itemno<<" ";
  cout<<Front->Itemname<<"Deleted";
  Front=Front->Link;
  delete Temp;
  if (Front NULL)
        Rear=NULL;
  }
  else
      cout<<"Queue Empty..";
}
```

Write a function QUEINS( ) in C++ to insert an element in a dynamically allocated Queue containing nodes of the following given structure:

```
struct Node
{
   int PId ;              //Product Id char Pname [20] ;
   NODE *Next ;
} ;
```

Ans:

```
class Queue
{ Node *Front, *Rear;
public:
  QUEUE( )
        //Constructor to initialize Front and Rear
{                Front = NULL; Rear = NULL;
    }
void QUEINS( ); //Function to insert a node void QUEDEL( ); //Function to delete
 a node void QUEDISP( );//Function to display nodes
  ~Queue( ); //Destructor to delete all nodes
} ;
void Queue::QUEINS( )
{ Node *Temp; Temp = new Node; cin>>Temp->PId;
```

```
     gets(Temp->Pname);
             //Or cin>>Temp->Pname;
                //cin.getline(Temp->Pname); Temp->Next = NULL;
     if (Rear = = NULL)
     { Front = Temp; Rear = Temp;
      }
      else
      { Rear->Next = Temp; Rear = Temp;
       }
   }
```

Write a function in C++ to insert an element into a dynamically allocated Queue where each node contains a name (of type string) as data.

Assume the following definition of THENODE for the same.

```
struct THENODE
   {
     char Name[20]; THENODE *Link;
   };
```

Solution:

```
struct THENODE
{ char Name[20]; THENODE *Link;
};
class Queue
{ THENODE *front,*rear; public:
   Queue( )
   {
    front = rear = NULL;
   }
   void Insert( ); void Delete( ); void Display( );
};
void Queue::Insert( )
{
    THENODE *ptr; ptr=new THENODE; if(ptr= = NULL)
    {
     cout<<"\nNo memory to create a
                        new node....";
     exit(1);
     }
    cout<<"\nEnter the name...."; gets(ptr->Name);
     ptr->Link=NULL;
    if(rear= = NULL)
           front=rear=ptr;
    else
     {
           Rear->Link=ptr; rear=ptr;
     }
}
```