Class declaration for Binary file functions

```
class student
{
   int roll;
   char name[20],subj[20];
   double marks;
   public:
      void inputdata()
      {
         cout<<"Roll ? "; cin>>roll;
         cout<<"Name ? "; gets(name);
         cout<<"Subject ? "; gets(subj);
         cout<<"Marks? "; cin>>marks;
      }
      void inputmarks()
      {
         cout<<"Marks? "; cin>>marks;
      }
      void showdata(){ cout<<roll<<','<<name<<','<<subj<<','<<marks<<endl; }
      int retroll()     { return roll; }
      double retmarks() { return marks; }
      char* retsub()    { return subj; }
      char* retname()   { return name; }
};
```

Binary Data File - Marks.dat

Roll	Name	Subject	Marks
1	ANITA	HINDI	85.5
2	KRISHNAN	FRENCH	73.0
3	SUDEEP	HINDI	79.0
4	PRAKASH	HINDI	96.5
5	MARY	HINDI	84.0
6	ZUBIN	FRENCH	95.5
7	MANOJ	HINDI	85.0
8	BIBHA	FRENCH	72.5
9	IQBAL	HINDI	87.5
10	BIPUL	HINDI	81.0
11	JOYDEEP	FRENCH	76.0
12	SANDEEP	FRENCH	82.5

```
void create1()
{
   student s;
   ofstream fout("Marks.dat",ios::binary);
   s.inputdata();
   fout.write((char*)&s,sizeof(s));
   fout.close();
}
```

One record is added in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void create1(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void create2()
{
   student s;
   char yes;
   ofstream fout("Marks.dat",ios::binary);
   do
   {
      s.inputdata();
      fout.write((char*)&s,sizeof(s));
      cout<<"Add More Record[Y/N]? "; cin>>yes;
   }
   while (yes=='Y' || yes=='y');
   fout.close();
}
```

At least one record is added in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void create2(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void create3(int n)
{
   student s;
   ofstream fout("Marks.dat",ios::binary);
   for (int k=0; k<n; k++)
   {
      s.inputdata();
      fout.write((char*)&s,sizeof(s));
   }
   fout.close();
}
```

`n` (`n` is passed as parameter to the function) records are added in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void create3(char* fname, int n)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void create4()
{
   student s;
   ofstream fout("Marks.dat",ios::binary);
   int n;
```

`n` (`n` is a local variable in the function) records are added in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void create4(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
    cout<<"Number Of Records"; cin>>n;
    for (int k=0; k<n; k++)
    {
       s.inputdata();
       fout.write((char*)&s,sizeof(s));
    }
    fout.close();
}
void append1()
{
    student s;
    ofstream fout("Marks.dat",ios::binary|ios::app);
    s.inputdata();
    fout.write((char*)&s,sizeof(s));
    fout.close();
}
void append2()
{
    student s;
    char yes;
    ofstream fout("Marks.dat",ios::binary|ios::app);
    do
    {
       s.inputdata();
       fout.write((char*)&s,sizeof(s));
       cout<<"Add More Record[Y/N]? "; cin>>yes;
    }
    while (yes=='Y' || yes=='y');
    fout.close();
}
void append3(int n)
{
    student s;
    ofstream fout("Marks.dat",ios::binary|ios::app);
    for (int k=0; k<n; k++)
    {
       s.inputdata();
       fout.write((char*)&s,sizeof(s));
    }
    fout.close();
}
void append4()
{
    student s;
    ofstream fout("Marks.dat",ios::binary|ios::app);
    int n;
    cout<<"Number Of Records? ";
    cin>>n;
    for (int k=0; k<n; k++)
    {
       s.inputdata();
       fout.write((char*)&s,sizeof(s));
    }
    fout.close();
}
void display1()
{
    student s;
    ifstream fin("Marks.dat",ios::binary);
    fin.read((char*)&s,sizeof(s));
```

One record is appended in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void append1(char* fname)` and in file opening statement, `fname` replacing `"Marks.dat"`.

At least one record is appended in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void append2(char* fname)` and in file opening statement, `fname` replacing `"Marks.dat"`.

`n` (`n` is passed as parameter to the function) records are added in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void append3 (char* fname, int n)` and in file opening statement, `fname` replacing `"Marks.dat"`.

`n` (`n` is local variable in the function) records are appended in a binary file. If file name was passed as parameter to the function, changes will be in the function header `void append4 (char* fname)` and in file opening statement, `fname` replacing `"Marks.dat"`.

Function display all records stored in the binary file by using function `eof()`. If file name was passed as parameter to the function, changes will be in the function header `void display1(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
    while (!fin.eof())
    {
       s.showdata();
       fin.read((char*)&s,sizeof(s));
    }
    fin.close();
}
void display2()
{
    student s;
    ifstream fin("Marks.dat",ios::binary);
    fin.read((char*)&s,sizeof(s));
    while (fin)
    {
       s.showdata();
       fin.read((char*)&s,sizeof(s));
    }
    fin.close();
}
void display3()
{
    student s;
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       s.showdata();
    fin.close();
}
void display4()
{
    student s;
    ifstream fin("Marks.dat",ios::binary);
    fin.seekg(0,ios::end);
    int n=fin.tellg()/sizeof(s);
    fin.seekg(0);
    for (int k=0; k<n; k++)
    {
       fin.read((char*)&s,sizeof(s));
       s.showdata();
    }
    fin.close();
}
void search1(int roll)
{
    student s;
    int found=0;
    ifstream fin("Marks.dat",ios::binary);
    fin.read((char*)&s,sizeof(s));
    while (fin && found==0)
       if (s.retroll()==roll)
       {
          cout<<"Record Found\n";
          s.showdata();
          found=1;
       }
       else
          fin.read((char*)&s,sizeof(s));
    fin.close();
    if (found==0)
       cout<<"No Record Found With Roll="<<roll<<endl;
}
```

All records are read (and displayed on the screen) from binary file by using file object `fin`. If file name was passed as parameter to the function, changes will be in the function header **`void`** `display2(`**`char`**`* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

All records are read (and displayed on the screen) from binary file by using stream class member function `read().` If file name was passed as parameter to the function, changes will be in the function header **`void`** `display3(`**`char`**`* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

All records are read (and displayed on the screen) from binary file by using stream class member functions `seekg()` and `tellg().` If file name was passed as parameter to the function, changes will be in the function header **`void`** `display4(`**`char`**`* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

Function searches for a roll number (**`int`** `roll`) passed as parameter to the function. If file name was passed as parameter to the function, changes will be in the function header **`void`** `search1(`**`char`**`* fname, `**`int`**` roll)` and in file opening statement `"Marks.dat"` replaced by `fname`. Flag variable `found` is used to terminate the loop when search is successful.

```
void search2()
{
   student s;
   int roll,found=0;
   cout<<"Roll To Search? "; cin>>roll;
   ifstream fin("Marks.dat",ios::binary);
   fin.read((char*)&s,sizeof(s));
   while (fin && found==0)
      if (s.retroll()==roll)
      {
         cout<<"Record Found\n";
         s.showdata();
         found=1;
      }
      else
         fin.read((char*)&s,sizeof(s));
   fin.close();
   if (found==0)
      cout<<"No Record Found With Roll="<<roll<<endl;
}
```

Function searches for a roll number where **int** roll is a local variable in the function. If file name was passed as parameter to the function, changes will be in the function header **void** search2(**char*** fname) and in file opening statement "Marks.dat" replaced by fname. Flag variable found is used to terminate the loop when search is successful.

```
void search3(char name[])
{
   student s;
   int found=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (stricmp(s.retname(),name)==0)
      {
         s.showdata();
         found=1;
      }
   fin.close();
   if (found==0)
      cout<<"No Record Found With Name="<<name<<endl;
}
```

Function searches for a name (**char** name[]) passed as parameter to the function. If file name was passed as parameter to the function, changes will be in the function header **void** search3(**char*** fname, **char** name[]) and in file opening statement "Marks.dat" replaced by fname. Name may not be unique hence **while** loop reads all the records.

```
void search4()
{
   student s;
   int found=0;
   char name[20];
   cout<<"Name To Search? "; gets(name);
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (stricmp(s.retname(),name)==0)
      {
         s.showdata();
         found=1;
      }
   fin.close();
   if (found==0)
      cout<<"No Record Found With Name="<<name<<endl;
}
```

Function searches for a name where **char** name[20] is local to the function. If file name was passed as parameter to the function, changes will be in the function header **void** search4(**char*** fname) and in file opening statement "Marks.dat" replaced by fname. Name may not be unique hence **while** loop reads all the records.

```
void search5(char sub[])
{
   student s;
   int found=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (stricmp(s.retsub(),sub)==0)
      {
         s.showdata(); found=1;
      }
```

Function searches for a subject (**char** sub[]) passed as parameter to the function. If file name was passed as parameter to the function, changes will be in the function header **void** search5(**char*** fname, **char** sub[]) and in file opening statement "Marks.dat" replaced by fname. Subject name may not be unique hence **while** loop reads all the records.

```
    fin.close();
    if (found==0)
       cout<<"No Record Found With Subject="<<sub<<endl;
}
void search6()
{
    student s;
    int found=0;
    char sub[20];
    cout<<"Subject To Search? "; gets(sub);
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       if (stricmp(s.retsub(),sub)==0)
       {
          s.showdata();
          found=1;
       }
    fin.close();
    if (found==0)
       cout<<"No Record Found With Subject="<<sub<<endl;
}
```

Function searches for a subject where **char** sub[20] is local to the function. If file name was passed as parameter to the function, changes will be in the function header **void** search6(**char*** fname) and in file opening statement "Marks.dat" replaced by fname. Subject name may not be unique hence **while** loop reads all the records.

```
int countrec1()
{
    student s;
    int count=0;
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       count++;
    fin.close();
    return count;
}
```

Function counts the number of records in a binary file by reading every record from the file. If file name was passed as parameter to the function, changes will be in the function header **void** countrec1(**char*** fname) & in file opening statement "Marks.dat" replaced by fname.

```
int countrec2()
{
    ifstream fin("Marks.dat",ios::binary);
    fin.seekg(0,ios::end);
    int n=fin.tellg()/sizeof(student);
    fin.close();
    return n;
}
```

Function counts the number of records in a binary file by using stream class member functions seekg() & tellg(). If file name was passed as parameter to the function, changes will be in the function header **void** countrec2(**char*** fname) and in file opening statement fname replacing "Marks.dat".

```
int counthindi()
{
    student s;
    int count=0;
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       if (strcmp(s.retsub(),"HINDI")==0)
          count++;
    fin.close();
    return count;
}
```

Function counts the number of students having **Hindi**. If file name was passed as parameter to the function, changes will be in the function header **void** counthindi(**char*** fname) and in file opening statement "Marks.dat" replaced by fname. If the return value of the function is **void** then cout<<count replacing **return** count.

```
int countfrench()
{
    student s;
    int count=0;
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       if (strcmp(s.retsub(),"FRENCH")==0)
          count++;
    fin.close();
    return count;
}
```

Function counts the number of students having **French**. If file name was passed as parameter to the function, changes will be in the function header **void** countfrench(**char*** fname) and in file opening statement "Marks.dat" replaced by fname. If the return value of the function is **void** then cout<<count replacing **return** count.

```
void counthindifrench1()
{
   student s;
   int counthi=0, countfr=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (strcmp(s.retsub(),"HINDI")==0)
         counthi++;
      else
         countfr++;
   fin.close();
   cout<<"Number Of Hindi Students ="<<counthi<<endl;
   cout<<"Number Of French Students="<<countfr<<endl;
}
```

Function counts the number of students having **Hindi** and **French**. If file name was passed as parameter to the function, changes will be in the function header `void counthindifrench1 (char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void counthindifrench2()
{
   student s;
   int counthi=0, countfr=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (strcmp(s.retsub(),"FRENCH")==0)
         countfr++;
      else
         counthi++;
   fin.close();
   cout<<"Number Of Hindi Students ="<<counthi<<endl;
   cout<<"Number Of French Students="<<countfr<<endl;
}
```

Function counts the number of students having **Hindi** and **French**. If file name was passed as parameter to the function, changes will be in the function header `void counthindifrench2 (char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void counthindifrench3()
{
   student s;
   int counthi=0, countfr=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (strcmp(s.retsub(),"FRENCH")==0)
         countfr++;
      else
      if (strcmp(s.retsub(),"HINDI")==0)
         counthi++;
   fin.close();
   cout<<"Number Of Hindi Students ="<<counthi<<endl;
   cout<<"Number Of French Students="<<countfr<<endl;
}
```

Function counts the number of students having **Hindi** and **French**. If file name was passed as parameter to the function, changes will be in the function header `void counthindifrench3 (char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void counthindifrench4()
{
   student s;
   int counthi=0, countfr=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
   {
      if (strcmp(s.retsub(),"FRENCH")==0)
         countfr++;
      if (strcmp(s.retsub(),"HINDI")==0)
         counthi++;
   }
   fin.close();
   cout<<"Number Of Hindi Students ="<<counthi<<endl;
   cout<<"Number Of French Students="<<countfr<<endl;
}
```

Function counts the number of students having **Hindi** and **French**. If file name was passed as parameter to the function, changes will be in the function header `void counthindifrench4 (char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
double hindiaverage()
{
   student s;
   int count=0;
   double sum=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (strcmp(s.retsub(),"HINDI")==0)
      {
         sum+=s.retmarks();
         count++;
      }
   fin.close();
   double avg=sum/count;
   return avg;
}
```

Function calculates average marks obtained by **Hindi** students. If file name was passed as parameter to the function, changes will be in the function header `void hindiaverage(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`. If the return value of the function is `void` then `cout<<avg` replacing `return avg.`

```
double frenchaverage()
{
   student s;
   int count=0;
   double sum=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (strcmp(s.retsub(),"FRENCH")==0)
      {
         sum+=s.retmarks();
         count++;
      }
   fin.close();
   double avg=sum/count;
   return avg;
}
```

Function calculates average marks obtained by **French** students. If file name was passed as parameter to the function, changes will be in the function header `void frenchaverage(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`. If the return value of the function is `void` then `cout<<avg` replacing `return avg.`

```
void average()
{
   student s;
   int counthi=0,countfr=0;
   double sumhi=0,sumfr=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (strcmp(s.retsub(),"FRENCH")==0)
      {
         sumfr+=s.retmarks();
         countfr++;
      }
      else
      {
         sumhi+=s.retmarks();
         counthi++;
      }
   double avghi=sumhi/counthi, avgfr=sumfr/countfr;
   cout<<"Hindi Average ="<<avghi<<endl;
   cout<<"French Average="<<avgfr<<endl;
   fin.close();
}
```

Function calculates average marks obtained by **Hindi** students and **French** students. If file name was passed as parameter to the function, changes will be in the function header `void average(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
double hindimax()
{
   student s;
   double max=0;
   ifstream fin("Marks.dat",ios::binary);
   while (fin.read((char*)&s,sizeof(s)))
      if (s.retmarks()>max && strcmp(s.retsub(),"HINDI")==0)
         max=s.retmarks();
```

Function calculates highest marks obtained in **Hindi**. If file name was passed as parameter to the function, changes will be in the function header `void hindimax(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`. If the return value of the function is `void` then `cout<<max` replacing `return max.`

```
    fin.close();
    return max;
}
double frenchmax()
{
    student s;
    double max=0;
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       if (s.retmarks()>max && strcmp(s.retsub(),"FRENCH")==0)
          max=s.retmarks();
    fin.close();
    return max;
}
```

Function calculates highest marks obtained in **French**. If file name was passed as parameter to the function, changes will be in the function header `void frenchmax(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`. If the return value of the function is `void` then `cout<<max` replacing `return max.`

```
void maxmarks()
{
    student s;
    double maxhi=0, maxfr=0;
    ifstream fin("Marks.dat",ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       if (s.retmarks()>maxhi && strcmp(s.retsub(),"HINDI")==0)
          maxhi=s.retmarks();
       else
       if (s.retmarks()>maxfr && strcmp(s.retsub(),"FRENCH")==0)
          maxfr=s.retmarks();
    fin.close();
    cout<<"Max in Hindi ="<<maxhi<<endl;
    cout<<"Max in French="<<maxfr<<endl;
}
```

Function calculates highest marks obtained in **Hindi** and in **French**. If file name was passed as parameter to the function, changes will be in the function header `void maxmarks(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void copyfile(char fname1[],char fname2[])
{
    student s;
    ofstream fout(fname1,ios::binary);
    ifstream fin(fname2,ios::binary);
    while (fin.read((char*)&s,sizeof(s)))
       fout.write((char*)&s,sizeof(s));
    fin.close();
    fout.close();
}
```

Function copies binary file `fname2` to another binary file `fnam1`. File names are passed as parameters to the function. File name `fname2` contains records of the type `student`.

```
void delrec1()
{
    student s;
    ifstream fin("Marks.dat",ios::binary);
    ofstream fout("temp.dat",ios::binary);
    int rno,found=0;
    cout<<"Roll For Deletion? "; cin>>rno;
    while (fin.read((char*)&s,sizeof(s)))
       if (s.retroll()!=rno)
          fout.write((char*)&s,sizeof(s));
       else
          found=1;
    if (found==1)
       cout<<"Record Deleted From The File\n";
    else
       cout<<"Record Could Not Be Located In The File\n";
    fin.close();
    fout.close();
    remove("Marks.dat");
    rename("temp.dat","Marks.dat");
}
```

A record is deleted from a binary file using roll. Roll is a local variable in the function. If file name was passed as parameter to the function, changes will be in the function header `void delrec1 (char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void delrec2(int rno)
{
   student s;
   ifstream fin("Marks.dat",ios::binary);
   ofstream fout("temp.dat",ios::binary);
   int found=0;
   while (fin.read((char*)&s,sizeof(s)))
      if (s.retroll()!=rno)
         fout.write((char*)&s,sizeof(s));
      else
         found=1;
   if (found==1)
      cout<<"Record Deleted From The File\n";
   else
      cout<<"Record Could Not Be Located In The File\n";
   fin.close(); fout.close();
   remove("Marks.dat");
   rename("temp.dat","Marks.dat");
}
```

A record is deleted from a binary file using roll. Roll is passed as parameter to the function. If file name was passed as parameter to the function, changes will be in the function header **void** delrec2(**char*** fname) and in file opening statement "Marks.dat" replaced by fname.

```
void editrec1()
{
   student s;
   ifstream fin("Marks.dat",ios::binary);
   ofstream fout("temp.dat",ios::binary);
   int rno,found=0;
   cout<<"Roll For Updation? "; cin>>rno;
   while (fin.read((char*)&s,sizeof(s)))
   {
      if (s.retroll()==rno)
      {
         s.inputmarks();
         cout<<"Record Updated In The File\n";
         found=1;
      }
      fout.write((char*)&s,sizeof(s));
   }
   if (found==0)
      cout<<"Record Could Not Be Located In The File\n";
   fin.close(); fout.close();
   remove("Marks.dat");
   rename("temp.dat","Marks.dat");
}
```

A record is updated in a binary file using roll and temporary file. Roll is local variable in the function. If file name was passed as parameter to the function, changes will be in the function header **void** editrec1(**char*** fname) and in file opening statement "Marks.dat" replaced by fname.

```
void editrec2()
{
   student s;
   int rno,found=0;
   cout<<"Roll For Updation? "; cin>>rno;
   fstream f("Marks.dat",ios::binary|ios::in|ios::out);
   while (f.read((char*)&s,sizeof(s)) && found==0)
      if (s.retroll()==rno)
      {
         s.inputmarks();
         f.seekp(-sizeof(s),ios::cur);
         f.write((char*)&s,sizeof(s));
         cout<<"Record Updated In The File\n";
         found=1;
      }
   if (found==0)
      cout<<"Record Could Not Be Located In The File\n";
   f.close();
}
```

A record is updated in a binary file using roll and without using temporary file (opening the file in input / output mode). Roll is local variable in the function. If file name was passed as parameter to the function, changes will be in the function header **void** editrec2(**char*** fname) and in file opening statement fname replacing "Marks.dat".

```
void editrec3(int rno)
{
   student s;
   ifstream fin("Marks.dat",ios::binary);
   ofstream fout("temp.dat",ios::binary);
   int found=0;
   while (fin.read((char*)&s,sizeof(s)))
   {
      if (s.retroll()==rno)
      {
         s.inputmarks(); found=1;
         cout<<"Record Updated In The File\n";
      }
      fout.write((char*)&s,sizeof(s));
   }
   if (found==0)
      cout<<"Record Could Not Be Located In The File\n";
   fin.close(); fout.close();
   remove("Marks.dat");
   rename("temp.dat","Marks.dat");
}
```

A record is updated in a binary file using roll and temporary file. Roll is passed as parameter to the function. If file name was passed as parameter to the function, changes will be in the function header `void editrec3(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void editrec4(int rno)
{
   student s;
   int found=0;
   fstream f("Marks.dat",ios::binary|ios::in|ios::out);
   while (f.read((char*)&s,sizeof(s)) && found==0)
      if (s.retroll()==rno)
      {
         s.inputmarks();
         f.seekp(-sizeof(s),ios::cur);
         f.write((char*)&s,sizeof(s));
         cout<<"Record Updated In The File\n";
         found=1;
      }
   if (found==0)
      cout<<"Record Could Not Be Located In The File\n";
   f.close();
}
```

A record is updated in a binary file using roll and without using temporary file (opening the file in input / output mode). Roll is passed as parameter to the function. If file name was passed as parameter to the function, changes will be in the function header `void editrec4(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void insertrecord()
{
   student a,s;
   ifstream fin("Marks.dat",ios::binary);
   ofstream fout("temp.dat",ios::binary);
   a.inputdata();
   int added=0;
   while (fin.read((char*)&s,sizeof(s)))
   {
      if (a.retroll()<s.retroll() && added==0)
      {
         fout.write((char*)&a,sizeof(a));
         added=1;
      }
      fout.write((char*)&s,sizeof(s));
   }
   if (added==0)
      fout.write((char*)&a,sizeof(a));
   fin.close(); fout.close();
   remove("Marks.dat");
   rename("temp.dat","Marks.dat");
}
```

Function adds a new record in file in a sorted position. A new record is added either in the beginning or somewhere in the middle or at the end depending on the value stored in the inputted roll number. If file name was passed as parameter to the function, changes will be in the function header `void insertrecord(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void sort1()
{
   fstream f("Marks.dat",ios::binary|ios::in|ios::out);
   f.seekg(0,ios::end);
   int size=sizeof(student),n=f.tellg()/size;
   student s1,s2;
   for (int k=1; k<n; k++)
      for (int x=0; x<n-k; x++)
      {
         f.seekg(x*size);
         f.read((char*)&s1,size);
         f.read((char*)&s2,size);
         if (s1.retroll()>s2.retroll())
         {
            f.seekp(x*size);
            f.write((char*)&s2,size);
            f.write((char*)&s1,size);
         }
      }
   f.close();
}
```

Function sorts records in a file using Bubble sort on roll. File is opened in an input/output mode. A pair of record is read from the file (`s1` reads the 1st record in the pair and `s2` is the 2nd record pair). If roll of `s1` > roll of `s2` then `s2` is stored in place of `s1` and `s1` is stored in place of `s2` (records are swapped). If file name was passed as parameter to the function, changes will be in the function header `void sort1(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
void sort2()
{
   fstream f("Marks.dat",ios::binary|ios::in|ios::out);
   f.seekg(0,ios::end);
   int size=sizeof(student),n=f.tellg()/size;
   student s1,s2;
   for (int k=1; k<n; k++)
      for (int x=0; x<n-k; x++)
      {
         f.seekg(x*size);
         f.read((char*)&s1,size);
         f.read((char*)&s2,size);
         if (strcmp(s1.retname(),s2.retname())>0)
         {
            f.seekp(x*size);
            f.write((char*)&s2,size);
            f.write((char*)&s1,size);
         }
      }
   f.close();
}
```

Function sorts records in a file using Bubble sort on name. File is opened in an input/output mode. A pair of record is read from the file (`s1` reads the 1st record in the pair and `s2` is the 2nd record pair). If name of `s1` > name of `s2` then `s2` is stored in place of `s1` and `s1` is stored in place of `s2` (records are swapped). If file name was passed as parameter to the function, changes will be in the function header `void sort2(char* fname)` and in file opening statement `"Marks.dat"` replaced by `fname`.

```
int binarysearch(int rno)
{
   student s;
   ifstream fin("Marks.dat",ios::binary);
   fin.seekg(0,ios::end);
   int size=sizeof(s),n=fin.tellg()/size;
   int lb=0,ub=n-1,found=0,mid;
   while (lb<=ub && found==0)
   {
      mid=(lb+ub)/2;
      fin.seekg(mid*size);
      fin.read((char*)&s,size);
      if (rno>s.retroll()) lb=mid+1;
      if (rno<s.retroll()) ub=mid-1;
      if (rno==s.retroll()) found=1;
   }
   fin.close();
   return found;
}
```

Function searches for a roll number in a file sorted on roll number by applying binary search. Roll number is passed as parameter to the function. If return value of the function was `void` then `return found;` to be replace by

```
if (found==1)
   cout<<"Record Found In File";
else
   cout<<"Record Not Found";
```

Binary Data File-HiMarks.dat

Roll	Name	Subject	Marks
1	ANITA	FRENCH	85.5
2	SAURAV	FRENCH	67.0
4	KAVITA	FRENCH	71.5
5	KRISHNAN	FRENCH	73.0
6	SUDEEP	FRENCH	79.0
9	PRAKASH	FRENCH	96.5
13	MARY	FRENCH	84.0
15	ANUPA	FRENCH	80.5
18	SMITA	FRENCH	83.0

Binary Data File-FrMarks.dat

Roll	Name	Subject	Marks
3	ZUBIN	HINDI	95.5
7	HUSSAIN	HINDI	62.5
8	MARIAM	HINDI	78.5
10	MANOJ	HINDI	85.0
12	BIBHA	HINDI	72.5
14	IQBAL	HINDI	87.5
16	BIPUL	HINDI	81.0
17	JOYDEEP	HINDI	76.0
19	SANDEEP	HINDI	82.5

Merged File - NewMarks.dat

Roll	Name	Subject	Marks
1	ANITA	FRENCH	85.5
2	SAURAV	FRENCH	67.0
3	ZUBIN	HINDI	95.5
4	KAVITA	FRENCH	71.5
5	KRISHNAN	FRENCH	73.0
6	SUDEEP	FRENCH	79.0
7	HUSSAIN	HINDI	62.5
8	MARIAM	HINDI	78.5
9	PRAKASH	FRENCH	96.5
10	MANOJ	HINDI	85.0
12	BIBHA	HINDI	72.5
13	MARY	FRENCH	84.0
14	IQBAL	HINDI	87.5
15	ANUPA	FRENCH	80.5
16	BIPUL	HINDI	81.0
17	JOYDEEP	HINDI	76.0
18	SMITA	FRENCH	83.0
19	SANDEEP	HINDI	82.5

```
void mergefile()
{
   student s1,s2;
   ifstream f1("FrMarks.dat",ios::binary);
   ifstream f2("HiMarks.dat",ios::binary);
   ofstream fout("NewMarks.dat",ios::binary);
   int size=sizeof(student);
   f1.read((char*)&s1,size);
   f2.read((char*)&s2,size);
   while (f1 && f2)
      if (s1.retroll()<s2.retroll())
      {
         fout.write((char*)&s1,size);
         f1.read((char*)&s1,size);
      }
      else
      {
         fout.write((char*)&s2,size);
         f2.read((char*)&s2,size);
      }
   while (f1)
   {
      fout.write((char*)&s1,size);
      f1.read((char*)&s1,size);
   }
   while (f2)
   {
      fout.write((char*)&s2,size);
      f2.read((char*)&s2,size);
   }
   f1.close();
   f2.close();
   fout.close();
}
```

Function merges 2 data files (Marks1.dat and Marks2.dat) sorted on roll and obtain a new data file (NewMarks.dat). Merging technique is similar to merging of two sorted arrays. A record is read from Marks1.dat (stored in s1) and Marks2.dat (stored in s2).

If roll of s1 < roll of s2 then s1 is written to NewMarks.dat and next record is read from Marks1.dat.

If roll of s1 >= roll of s2 then s2 is written to NewMarks.dat and next record is read from Marks2.dat.

If end of file is encountered for Marks1.dat then remaining records of Marks2.dat is written to NewMarks.dat.

If end of file is encountered for Marks2.dat then remaining records of Marks1.dat is written to NewMarks.dat.

Old Data File - OldMarks.dat

Roll	Name	Subject	Marks
1	ANITA	HINDI	85.5
2	KRISHNAN	FRENCH	73.0
3	SUDEEP	HINDI	79.0
4	PRAKASH	HINDI	96.5
5	MARY	HINDI	84.0
6	ZUBIN	FRENCH	95.5
7	MANOJ	HINDI	85.0
8	BIBHA	FRENCH	72.5
9	IQBAL	HINDI	87.5
10	BIPUL	HINDI	81.0
11	JOYDEEP	FRENCH	76.0
12	SANDEEP	FRENCH	82.5

New Data File - NewMarks.dat

Roll	Name	Subject	Marks
1	ANITA	HINDI	85.5
2	KRISHNAN	FRENCH	73.0
3	SUDEEP	HINDI	89.0
4	PRAKASH	HINDI	96.5
5	MARY	HINDI	85.0
6	ZUBIN	FRENCH	95.5
7	MANOJ	HINDI	85.0
8	BIBHA	FRENCH	74.0
9	IQBAL	HINDI	89.0
10	BIPUL	HINDI	81.0
11	JOYDEEP	FRENCH	76.0
12	SANDEEP	FRENCH	82.5

Temporary File-TempMarks.dat

Roll	Name	Subject	Marks
3	SUDEEP	HINDI	89.0
5	MARY	HINDI	85.0
8	BIBHA	FRENCH	74.0
9	IQBAL	HINDI	89.0

```
void updatefile()
{
   student s1,s2;
   ifstream f1("OldMarks.dat",ios::binary);
   ifstream f2("TempMarks.dat",ios::binary);
   ofstream fout("NewMarks.dat",ios::binary);
   int size=sizeof(student);
   f1.read((char*)&s1,size);
   f2.read((char*)&s2,size);
   while (f2)
      if (s1.retroll()==s2.retroll())
      {
         fout.write((char*)&s2,size);
         f1.read((char*)&s1,size);
         f2.read((char*)&s2,size);
      }
      else
      {
         fout.write((char*)&s1,size);
         f1.read((char*)&s1,size);
      }
   while (f1)
   {
      fout.write((char*)&s1,size);
      f1.read((char*)&s1,size);
   }
   f1.close();
   f2.close();
   fout.close();
}
```

Function updates records of an old file (OldMarks.dat) by using a temporary file (TempMarks.dat) to obtain a new file (NewMarks.dat). A record is read from Marks1.dat (stored in s1) and Marks2.dat (stored in s2).
If roll of s1 == roll of s2 then s2 is written to NewMarks.dat and next record is read from Marks1.dat and Marks2.dat.
If roll of s1 < roll of s2 then s1 is written to NewMarks.dat and next record is read from Marks1.dat.
If end of file is encountered for Marks2.dat then remaining records of Marks1.dat is written to NewMarks.dat.

Text File Functions

```
void createtextfile1()
{
   ofstream fout("News.txt");
   char string[80], yes;
   do
   {
      cout<<"Enter a line of text? \n"; gets(string);
      fout<<string<<endl;
      cout<<"Add More line[Y/N]? "; cin>>yes;
   }
   while (yes=='Y' || yes=='y');
   fout.close();
}
void createtextfile2(int n)
{
   ofstream fout("News.txt");
   char string[80];
   for (int k=1; k<=n; k++)
   {
      cout<<"Enter a line of text? \n"; gets(string);
      fout<<string<<endl;
   }
   fout.close();
}
void appendtextfile1()
{
   ofstream fout("News.txt",ios::app);
   char string[80], yes;
   do
   {
      cout<<"Enter a line of text? \n"; gets(string);
      fout <<string<<endl;
      cout<<"Add More line[Y/N]? "; cin>>yes;
   }
   while (yes=='Y' || yes=='y');
   fout.close();
}
void appendtextfile2(int n)
{
   ofstream fout("News.txt",ios::app);
   char string[80];
   for (int k=1; k<=n; k++)
   {
      cout<<"Enter a line of text? \n"; gets(string);
      fout<<string<<endl;
   }
   fout.close();
}
void displaytextfile1()
{
   ifstream fin("News.txt");
   char ch;
   while (fin.get(ch))
      cout<<ch;
   fin.close();
}
void displaytextfile2()
{
   ifstream f("News.txt");
   char word[20];
   while (f>>word)
```

Function displays text file character by character.

Function displays text file word by word. `f>>word` can be replaced by `f.getline(word,20,' ')`. Function assumes that there are no new line characters (`'\n'`) present in the text file.

```
      cout<<word<<' ';
   f.close();
}
void displaytextfile3()
{
   ifstream fin("News.txt");
   char string[80];
   while (fin.getline(string, 80))
      cout<<string<<endl;
   fin.close();
}
int countupper()
{
   ifstream fin("News.txt");
   char ch;
   int count=0;
   while (fin.get(ch))
      if (ch>='A' && ch<='Z')
         count++;
   fin.close();
   return count;
}
int countlower()
{
   ifstream fin("News.txt");
   char ch;
   int count=0;
   while (fin.get(ch))
      if (ch>='a' && ch<='z')
         count++;
   fin.close();
   return count;
}
int countdigits()
{
   ifstream fin("News.txt");
   char ch;
   int count=0;
   while (fin.get(ch))
      if (ch>='0' && ch<='9')
         count++;
   fin.close();
   return count;
}
int countspecial()
{
   ifstream fin("News.txt");
   char ch;
   int count=0;
   while (fin.get(ch))
      if (isalnum(ch)==0)
         count++;
   fin.close();
   return count;
}
int countletter()
{
   ifstream fin("News.txt");
   char ch;
   int count=0;
   while (fin.get(ch))
      if (ch>='A' && ch<='Z' || ch>='a' && ch<='z')
```

Function assumes that each line of text contains maximum 79 characters. Each line of text is separated by new line character (`'\n'`).

`ch>='A' && ch<='Z'` can be replaced with `isupper(ch)`. If the return value of the function is **`void`** then **`return`** `count` will be replaced by `cout<<count`.

`ch>='a' && ch<='z'` can be replaced with `islower(ch)`. If the return value of the function is **`void`** then **`return`** `count` will be replaced by `cout<<count`.

`ch>='0' && ch<='9'` can be replaced with `isdigit(ch)`. If the return value of the function is **`void`** then **`return`** `count` will be replaced by `cout<<count`.

If the return value of the function is **`void`** then **`return`** `count` will be replaced by `cout<<count`.

`ch>='A' && ch<='Z' || ch>='a' && ch<='z'` can be replaced with `isalpha(ch)`. If the return value of the function is **`void`** then **`return`** `count` will be replaced by `cout<<count`.

```
         count++;
   fin.close();
   return count;
}
int countalphanumeric()
{
   ifstream fin("News.txt");
   char ch;
   int count=0;
   while (fin.get(ch))
      if (ch>='A' && ch<='Z' || ch>='a' && ch<='z' || ch>='0' && ch<='9')
         count++;
   fin.close();
   return count;
}
void countupperlower()
{
   ifstream fin("News.txt");
   int up=0, lo=0;
   char ch;
   while (fin.get(ch))
      if (ch>='A' && ch<='Z')
         up++;
      else
      if (ch>='a' && ch<='z')
         lo++;
   fin.close();
   cout<<"No. of uppercase="<<up<<endl<<"No. of lowercase="<<lo<<endl;
}
void countdiffchar1()
{
   ifstream fin("News.txt");
   int up=0, lo=0, di=0, sp=0;
   char ch;
   while (fin.get(ch))
      if (ch>='A' && ch<='Z')
         up++;
      else
      if (ch>='a' && ch<='z')
         lo++;
      else
      if (ch>='0' && ch<='9')
         di++;
      else
         sp++;
   fin.close();
   cout<<"No. of uppercase="<<up<<endl<<"No. of lowercase="<<lo<<endl;
   cout<<"No. digits="<<di<<endl<<"No. of special character="<<sp<<endl;
}
void countdiffchar2()
{
   ifstream fin("News.txt");
   int up=0, lo=0, di=0, sp=0;
   char ch;
   while (fin.get(ch))
      switch (isalnum(ch))
      {
         case 0: sp++; break;
         case 2: di++; break;
         case 4: up++; break;
         case 8: lo++; break;
      }
```

`ch>='A' && ch<='Z' || ch>='a' && ch<='z' || ch>='0' && ch<='9'` can be replaced with `isalnum(ch)`. If the return value of the function is **`void`** then **`return`** `count` will be replaced by `cout<<count`.

`ch>='A' && ch<='Z'` can be replaced with `isupper(ch)`.
`ch>='a' && ch<='z'` can be replaced with `islower(ch)`.

`ch>='A' && ch<='Z'` can be replaced with `isupper(ch)`.
`ch>='a' && ch<='z'` can be replaced with `islower(ch)`.
`ch>='0' && ch<='9'` can be replaced with `isdigit(ch)`.

Return value of the function `isalnum()` is either `0` when `ch` is special character or `2` when `ch` is digit or `4` when `ch` is uppercase or `8` when `ch` is lowercase.

```
    fin.close();
    cout<<"No. of uppercase="<<up<<endl<<"No. of lowercase="<<lo<<endl;
    cout<<"No. digits="<<di<<endl<<"No. of special character="<<sp<<endl;
}
int countline(char* fname)
{
    ifstream fin(fname);
    char string[80];
    int count=0;
    while (fin.getline(string, 80))
        count++;
    fin.close();
    return count;
}
int countword1(char* fname)
{
    ifstream fin(fname);
    char ch;
    int count=0;
    while (fin.get(ch))
        if (ch==' ' || ch=='\n' || ch=='\t')
            count++;
    fin.close();
    return count;
}
int countword2(char* fname)
{
    ifstream fin (fname);
    char word[20];
    int count=0;
    while (fin>>word)
        count++;
    fin.close();
    return count;
}
void reversetextfileline1(char* fname)
{
    ifstream fin(fname);
    char string[80];
    while (fin.getline(string, 80))
    {
        for (int k=strlen(string)-1; k>=0; k--)
            cout<<string[k];
        cout<<endl;
    }
    fin.close();
}
void reversetextfileline2(char* fname)
{
    ifstream fin(fname);
    char string[80];
    while (fin.getline(string, 80))
    {
        strrev(string);
        cout<<string<<endl;
    }
    fin.close();
}
void copytextfile1(char* fname1, char* fname2)
{
    ofstream fout(fname1);
    ifstream fin(fname2);
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). If the return value of the function is **void** then **return** count will be replaced by cout<<count.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). If the return value of the function is **void** then **return** count will be replaced by cout<<count. Function assumes each word is separated by a blank (' ') / a new line ('\n') / a tab ('\t').

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). If the return value of the function is **void** then **return** count will be replaced by cout<<count. fin>>word can be replaced by fin.getline(word, 20, ' ').

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array).

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array).

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function copies text file character by character. fout<<ch can be replaced by fout.put(ch).

```
    char ch;
    while (fin.get(ch))
       fout<<ch;
    fout.close();
    fin.close();
}
void copytextfile2(char* fname1, char* fname2)
{
    ofstream fout(fname1);
    ifstream fin(fname2);
    char word[20];
    while (fin>>word)
       fout<<word<<' ';
    fout.close();
    fin.close();
}
void copytextfile3(char* fname1, char* fname2)
{
    ofstream fout(fname1);
    ifstream fin(fname2);
    char string[80];
    while (fin.getline(string, 80))
       fout<<string<<endl;
    fout.close();
    fin.close();
}
void copytextfileupper(char* fname1, char* fname2)
{
    ofstream fout(fname1);
    ifstream fin(fname2);
    char ch;
    while (fin.get(ch))
    {
       if (ch>='a' && ch<='z')
          ch=char(ch-32);
       fout<<ch;
    }
    fout.close();
    fin.close();
}
void copytextfilelower(char* fname1, char* fname2)
{
    ofstream fout(fname1);
    ifstream fin(fname2);
    char ch;
    while (fin.get(ch))
    {
       if (ch>='A' && ch<='Z')
          ch=char(ch+32);
       fout<<ch;
    }
    fout.close();
    fin.close();
}
void convertuppercase1(char* fname)
{
    ofstream fout(fname);
    ifstream fin("Temp.txt");
    char ch;
    while (fin.get(ch))
    {
       if (ch>='a' && ch<='z')
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function copies text file word by word. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`. Function assumes that there are no new line characters (`'\n'`) present in the text file.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function copies text file string by string. Function assumes that each line of text contains maximum 79 characters. Each line of text is separated by a new line character (`'\n'`).

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function copies text file character by character.
`if (ch>='a' && ch<='z') ch=char(ch-32);`
can be replaced by `ch=char(toupper(ch));`
`fout<<ch` can be replaced by `fout.put(ch)`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function copies text file character by character.
`if (ch>='A' && ch<='Z') ch=char(ch+32);`
can be replaced by `ch=char(tolower(ch));`
`fout<<ch` can be replaced by `fout.put(ch)`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function converts all letters to uppercase by using a temporary file.
`if (ch>='a' && ch<='z') ch=char(ch-32);`
can be replaced by `ch=char(toupper(ch));`
`fout<<ch` can be replaced by `fout.put(ch)`.

```
            ch=char(ch-32);
         fout<<ch;
      }
      fout.close();
      fin.close();
      remove(fname);
      rename("Temp.txt", fname);
}
void convertuppercase2(char* fname)
{
      fstream f(fname, ios::out|ios::in);
      char ch;
      while (f.get(ch))
      {
         if (ch>='a' && ch<='z')
         {
            ch=char(ch-32);
            f.seekg(-1, ios::cur);
            f<<ch;
         }
      }
      f.close();
}
void convertuppercase3(char* fname)
{
      fstream f(fname, ios::out|ios::in);
      char ch;
      while (f.get(ch))
      {
         ch=char(toupper(ch));
         f.seekg(-1, ios::cur);
         f<<ch;
      }
      f.close();
}
void convertlowercase1(char* fname)
{
      ofstream fout(fname);
      ifstream fin("Temp.txt");
      char ch;
      while (fin.get(ch))
      {
         if (ch>='A' && ch<='Z')
            ch=char(ch+32);
         fout<<ch;
      }
      fout.close();
      fin.close();
      remove(fname);
      rename("Temp.txt", fname);
}


void convertlowercase2(char* fname)
{
      fstream f(fname, ios::out|ios::in);
      char ch;
      while (f.get(ch))
         if (ch>='A' && ch<='Z')
         {
            ch=char(ch+32);
            f.seekg(-1, ios::cur);
```

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function converts all letters to uppercase without using a temporary file. `f<<ch` can be replaced by `f.put(ch)`.

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function converts all letters to uppercase without using a temporary file. `f<<ch` can be replaced by `f.put(ch)`.

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function converts all letters to lowercase by using a temporary file.
`if (ch>='A' && ch<='Z') ch=char(ch+32);`
can be replaced by `ch=char(toupper(ch));`
`fout<<ch` can be replaced by `fout.put(ch)`.

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function converts all letters to lowercase without using a temporary file. `f<<ch` can be replaced by `f.put(ch)`.

```
            f<<ch;
         }
      f.close();
}
void convertlowercase3(char* fname)
{
   fstream f(fname, ios::out|ios::in);
   char ch;
   while (f.get(ch))
   {
      ch=char(tolower(ch));
      f.seekg(-1, ios::cur);
      f<<ch;
   }
   f.close();
}
void togglecase1(char* fname)
{
   ofstream fout(fname);
   ifstream fin("Temp.txt");
   char ch;
   while (fin.get(ch))
   {
      if (ch>='A' && ch<='Z')
         ch=char(ch+32);
      else
      if (ch>='a' && ch<='z')
         ch=char(ch-32);
      fout<<ch;
   }
   fout.close(); fin.close();
   remove(fname);
   rename("Temp.txt", fname);
}
void togglecase2(char* fname)
{
   fstream f(fname, ios::out|ios::in);
   char ch;
   while (f.get(ch))
      if (ch>='A' && ch<='Z')
      {
         f.seekg(-1, ios::cur);
         f<<char(ch+32);
      }
      else
      if (ch>='a' && ch<='z')
      {
         f.seekg(-1, ios::cur);
         f<<char(ch-32);
      }
   f.close();
}
void togglecase3(char* fname)
{
   char ch;
   fstream f(fname, ios::out|ios::in);
   while (f.get(ch))
      if (ch>='A' && ch<='Z')
      {
         f.seekg(-1, ios::cur);
         f<<char(tolower(ch));
      }
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function converts all letters to lowercase without using a temporary file. `f<<ch` can be replaced by `f.put(ch)`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function changes uppercase to lowercase and vice-versa by using a temporary file. `fout<<ch` can be replaced by `fout.put(ch)`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function changes uppercase to lowercase and vice-versa without using a temporary file. `f<<ch` can be replaced by `f.put(ch)`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function changes uppercase to lowercase and vice-versa without using a temporary file.

```
      else
      if (ch>='a' && ch<='z')
      {
         f.seekg(-1, ios::cur);
         f<<char(toupper(ch));
      }
   f.close();
}
void togglecase4(char* fname)
{
   char ch;
   fstream f(fname, ios::out|ios::in);
   while (f.get(ch))
      switch (isalpha(ch))
      {
         case 4: f.seekg(-1, ios::cur); f<<char(tolower(ch)); break;
         case 8: f.seekg(-1, ios::cur); f<<char(toupper(ch)); break;
      }
   f.close();
}
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function changes uppercase to lowercase and vice-versa without using a temporary file.

```
int countword3(char* fname)
{
   ifstream fin(fname);
   char word[20];
   int count=0;
   while (fin>>word)
      if (word[0]=='A' || word[0]=='a')
         count++;
   fin.close();
   return count;
}
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of words starting with A/a. If the return value of the function is **void** then **return** count will be replaced by cout<<count. fin>>word can be replaced by fin.getline(word, 20, ' ').

```
int countword4(char* fname)
{
   ifstream fin(fname);
   char word[20];
   int count=0;
   while (fin>>word)
   {
      int found=0;
      for (int k=0; word[k]!='\0' && found==0; k++)
         if (word[k]=='E' || word[k]=='e' || word[k]=='I' || word[k]=='i')
         {
            found=1;
            count++;
         }
   }
   fin.close();
   return count;
}
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of words containing E/I/e/i. If the return value of the function is **void** then **return** count will be replaced by cout<<count. fin>>word can be replaced by fin.getline(word, 20, ' ').

```
int countword5(char* fname)
{
   ifstream fin(fname);
   char word[20];
   int count=0;
   while (fin>>word)
   {
      char ch=word[0];
      if (ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U' ||
          ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
         count++;
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of words starting with vowel. If the return value of the function is **void** then **return** count will be replaced by cout<<count. fin>>word can be replaced by fin.getline(word, 20, ' ').

```
    }
    fin.close();
    return count;
}
int countword6(char* fname)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    while (fin>>word)
    {
       char ch=char(toupper(word[0]));
       if (ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U')
          count++;
    }
    fin.close();
    return count;
}
int countword7(char* fname)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    while (fin>>word)
    {
       char ch=char(tolower(word[0]));
       if (ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
          count++;
    }
    fin.close();
    return count;
}
int countword8(char* fname)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    while (fin>>word)
       if (stricmp("the", word)==0)
          count++;
    fin.close();
    return count;
}
int countword9(char* fname)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    while (fin>>word)
    {
       strlwr(word);
       if (strcmp("the", word)==0)
          count++;
    }
    fin.close();
    return count;
}
int countword10(char* fname)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
```

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of words starting with `vowel`. If the return value of the function is **void** then **return** `count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of words starting with `vowel`. If the return value of the function is **void** then **return** `count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of times the string `"the"` is present in the file. If the return value of the function is **void** then **return** `count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`. Function `stricmp()` can be replaced by `strcmpi()`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of times the string `"the"` is present in the file. If the return value of the function is **void** then **return** `count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

Filename is a string, so the parameter is either **char*** (pointer) or **char[]** (array). Function counts number of times the string `"the"` is present in the file. If the return value of the function is **void** then **return** `count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

```
    while (fin>>word)
    {
        strupr(word);
        if (strcmp("THE", word)==0)
            count++;
    }
    fin.close();
    return count;
}
int countword11(char* fname, char* str)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    strupr(str);
    while (fin>>word)
    {
        strupr(word);
        if (strcmp(str, word)==0)
            count++;
    }
    fin.close();
    return count;
}
```

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function counts number of times the word contained in the string `str` is present in the file. If the return value of the function is `void` then `return count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

```
int countword12(char* fname, char* str)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    strlwr(str);
    while (fin>>word)
    {
        strlwr(word);
        if (strcmp(str, word)==0)
            count++;
    }
    fin.close();
    return count;
}
```

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function counts number of times the word contained in the string `str` is present in the file. If the return value of the function is `void` then `return count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

```
int countword13(char* fname, char* str)
{
    ifstream fin(fname);
    char word[20];
    int count=0;
    while (fin>>word)
        if (stricmp(str, word)==0)
            count++;
    fin.close(); return count;
}
```

Filename is a string, so the parameter is either `char*` (pointer) or `char[]` (array). Function counts number of times the word contained in the string `str` is present in the file. If the return value of the function is `void` then `return count` will be replaced by `cout<<count`. `fin>>word` can be replaced by `fin.getline(word, 20, ' ')`.

```
Text File  -  Marks.txt
Name     Subject  Marks
ANITA    HINDI     85.5
KRISHNAN FRENCH    73.0
SUDEEP   HINDI     79.0
PRAKASH  HINDI     96.5
MARY     HINDI     84.0
ZUBIN    FRENCH    95.5
BIBHA    FRENCH    72.5
IQBAL    HINDI     87.5
BIPUL    HINDI     81.0
JOYDEEP  FRENCH    76.0
```

```
void countstudent()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks;
   int hin=0, fre=0;
   while (fin>>name>>subject>>marks)
      if (strcmp(subject, "HINDI")==0)
         hin++;
      else
         fre++;
   fin.close();
   cout<<"No. of Hindi student ="<<hin<<endl;
   cout<<"No. of French student="<<fre<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function counts number of student having Hindi and number of students having French.

```
void calculateaverage1()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks;
   int hin=0, fre=0;
   double sumh=0, sumf=0;
   while (fin>>name>>subject>>marks)
      if (strcmp(subject, "HINDI")==0)
      {
         sumh+=marks;
         hin++;
      }
      else
      {
         sumf+=marks;
         fre++;
      }
   fin.close();
   double avgh=sumh/hin, avgf=sumf/fre;
   cout<<"Average marks in Hindi ="<<avgh<<endl;
   cout<<"Average marks in French="<<avgf<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function calculates average marks obtained in Hindi in French.

```
void maxmarks1()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks, max=0;
   while (fin>>name>>subject>>marks)
      if (strcmp(subject, "HINDI")==0 && marks>max)
         max=marks;
   fin.close();
   cout<<"Highest Marks in Hindi="<<max<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function displays highest marks obtained in Hindi.

```
void maxmarks2()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks, max=0;
   while (fin>>name>>subject>>marks)
      if (strcmp(subject, "FRENCH")==0 && marks>max)
         max=marks;
   fin.close();
   cout<<"Highest Marks in French="<<max<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function displays highest marks obtained in French.

```
void maxmarks3()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks, max=0;
   while (fin>>name>>subject>>marks)
      if (strcmp(subject,"FRENCH")==0)
         if (marks>max)
            max=marks;
   fin.close();
   cout<<"Highest Marks in French="<<max<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function displays highest marks obtained in French.

```
void maxmarks4()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks, maxh=0, maxf=0;
   while (fin>>name>>subject>>marks)
   {
      if (strcmp(subject, "HINDI")==0 && marks>maxh)
         maxh=marks;
      if (strcmp(subject, "FRENCH")==0 && marks>maxf)
         maxf=marks;
   }
   fin.close();
   cout<<"Highest Marks in Hindi ="<<maxh<<endl;
   cout<<"Highest Marks in French="<<maxf<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function displays highest marks obtained in Hindi and French.

```
void maxmarks5()
{
   ifstream fin("Marks.txt");
   char name[15], subject[10];
   double marks, maxh=0, maxf=0;
   while (fin>>name>>subject>>marks)
      if (strcmp(subject, "HINDI")==0 && marks>maxh)
         maxh=marks;
      else
      if (strcmp(subject, "FRENCH")==0 && marks>maxf)
         maxf=marks;
   fin.close();
   cout<<"Highest Marks in Hindi ="<<maxh<<endl;
   cout<<"Highest Marks in French="<<maxf<<endl;
}
```

Text file Marks.txt contains student's name, subject name (Hindi / French) and marks (Hindi / French). Student's name only contains first name. Student's names and subject's names are in uppercase. Function displays highest marks obtained in Hindi and French.