TABLE OF CONTENTS

I. SYSTEM SPECIFICATION AT A GLANCE

Student report card system provides a user friendly solution for creating and maintaining student data and generating student report cards. System consists of a menu driven program to facilitate entry, modification and deletion of student record and generation of different reports. It provides permanent storage of data in files.

a) Program Modules:

*** Create student record**- A new student record can be created through this module. Student admission number, name, class, father's name, mother's name, batch year, date of birth and marks in different subjects can be entered into the system through this module.

***Display a student record**-This module takes the roll number of the student to be searched and displays the details if found

***Display student details of a class**-This module displays the details of all the students of a class in tabular form

***Modify student record**-Modifying an existing student record can be done through this module.

***Delete student record**-A student record can be deleted through this module.

***Student report card**-It provides the report card for an individual student. It contains student admission number, name, class, father's name, mother's name, batch year, date of birth and marks in different subjects, total, percentage and grade obtained by the student.

b) Files-System uses files to store data permanently on the disc

report.dat- This stores data of all students and is kept up-to-date with all additions, modifications and deletions of the student data.

c) Main inputs-System requires input from 2 sources namely from files and from user.

1) Input from the files

*Information about all existing students is read from file report.dat. This file is read for any further processing of existing students like searching, modification and deletion.

2) Inputs from user

* User interface is provided to create student data. A user is required to input information to create student data. User can also modify data of an existing student or delete a student record.

d) Main outputs-Outputs are in 2 forms, namely files and results.

1) Output to files

* Any new student records or modifications\deletions to existing student record are updated to the report.dat file.

2) Results- User can generate two types of results on screen.

***Class result**- Class result can be generated for any specific class chosen by user. It displays details of all the students of a class in tabular format.

***Student report card**- This report card of an individual student. It contains student admission number, name, class, batch year, date, and marks in all subjects, total, percentage and grade obtained by the student.

II.REQUIREMENT ANALYSIS

INPUT AND OUTPUT REQUIREMENTS

Required input of the system- System requires inputs from 2 sources, files and user

Input from files

System needs to retrieve student details created by user from time to time.Hence this data needs to read from permanent storage. A file can facilitate easy storage and retrieval.

User inputs

System should provide an interface for the user to create/maintain student data. Following inputs are required from the user for creating/modifying student information:

-Name of the student

-Class of the student

-Admission number of the student

-Marks obtained by the student in physics theory (out of 70)

-Marks obtained by the student in physics practical (out of 30)

-Marks obtained by the student in chemistry theory (out of 70)

-Marks obtained by the student in chemistry practical (out of 30)

-Marks obtained by the student in mathematics (out of 100)

-Marks obtained by the student in English (out of 100)

-Marks obtained by the student in computer science theory (out of 70)

-Marks obtained by the student in computer science practical (out of 30)

Required output of the system

Student information should be stored permanently to ensure retrieval from time to time for different purposes. A binary file should be used for this purpose to facilitate efficient storage and retrieval. Following details should be stored for each student.

*Admission number of a student

* Name of the student

*Class of the student

*Father's and Mother's name

*Date of birth

*Marks obtained by the student in physics theory

*Marks obtained by the student in physics practical

*Marks obtained by the student in chemistry theory

*Marks obtained by the student in chemisrty practical

*Marks obtained by the student in mathematics

*Marks obtained by the student in english

*Marks obtained by the student in computer science theory

*Marks obtained by the student in computer science practical

*Total marks obtained by the student

*Percentage of marks obtained by the student

*Grade obtained by the student

Results- System should provide two types of results.

Class result-System should provide a report that shows details of all students of a class selected by the user. The report should be in tabular format that provides following details:

*Admission number of a student

* Name of the student

*Class of the student

*Total marks obtained by the student in physics

*Total marks obtained by the student in chemistry

*Marks obtained by the student in mathematics

*Marks obtained by the student in english

*Total marks obtained by the student in computer science

*Aggregate marks obtained by the student

*Percentage of marks obtained by the student

*Grade obtained by the student

Student report card-System should provide report card for each individual student.

HARDWARE AND SOFTWARE REQUIREMENTS

Software requirement

*Turbo C++ 3.0 IDE is generally a simple software that does not have any complex hardware requirements

* Most C++ program can be run on a system having the following minimum requirements:

- A 32-bit processor

- A 256 MB RAM

- An Intel Pentium 3 processor

- At least 10 MB free space on the Hard disc

Organizing a computer with these basic requirements would not be of any problem at all

Hardware requirements

*The only requirement for Turbo C++ 3.0 IDE is

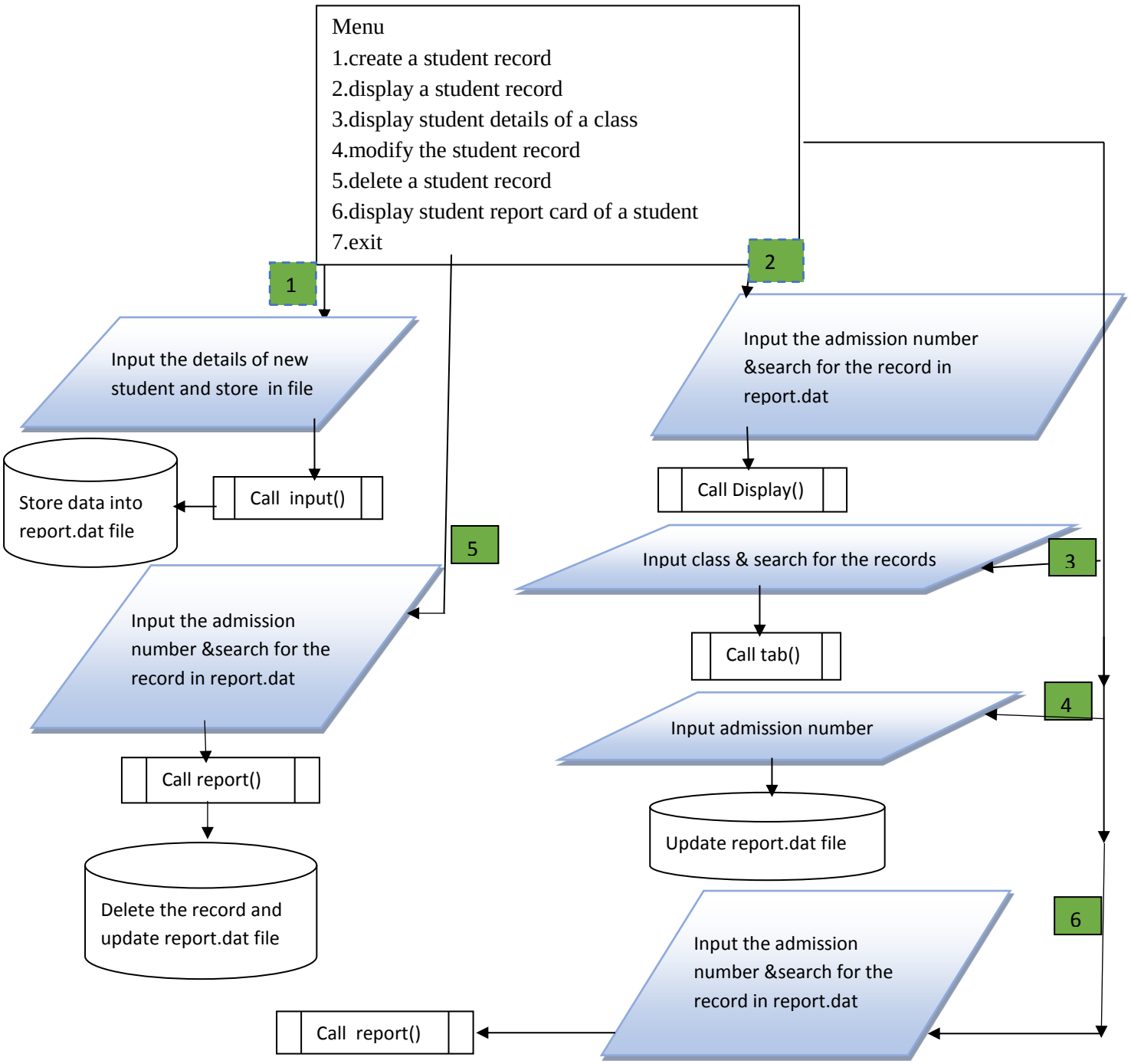
- A decent OS: Since this software has been released sometime in the early 1990's most of the present OS are compatible for running this software. Anything from windows Microsoft and Linux to Mac OS X would do

- But certain new Operating systems do not usually run the old DOS programs. In this case, Turbo C++ 3.0 IDE can be installed and run on the computer using an emulator software that emulates an IBM PC compatible computer running MS-DOS. An example is DOSBOX which is a command line program configured either by a set of command line arguments or by editing a plain text configuration file.

III. SYSTEM SPECIFICATION

System design

Block diagram – Shows all major parts and functions of the system and their relationship



Data Design

Student Report Card system is built around class student which contains all relevant information to generate student report card. Described below are the details of all the data members of class student. All the data members given below are private.

adno - It is an integer data member of class student that holds student admission number. This is the key to identify a student.

name- It is a character array of length 50 that holds student name.

sclass- It is an integer data member of class student that holds class of the student.

fname- It is a character array of length 50 that holds father name.

mname- It is a character array of length 50 that holds mother name.

batch-It is an integer data member of class student that holds current year for which the student report card is generated.

dob- Its of type structure dateofbirth.

marksp-It is an integer data member of class student that holds marks obtained by student in physics theory. Validation restricts this field to hold marks in range 0 to 70.

marksc-It is an integer data member of class student that holds marks obtained by the student in chemistry theory. Validation restricts this field to hold marks in range 0 to 70.

phyp- It is an integer data member of class student that holds marks obtained by the student in physics practical. Validation restricts this field to hold marks in range 0 to 30.

chemp- It is an integer data member of class student that holds marks obtained by the student in chemistry practical. Validation restricts this field to hold marks in range 0 to 30.

marksm- It is an integer data member of class student that holds marks obtained by the student in mathematics. Validation restricts this field to hold marks in range 0 to 100.

markscs- It is an integer data member of class student that holds marks obtained by the student in computer science theory. Validation restricts this field to hold marks in range 0 to 70.

csp- It is an integer data member of class student that holds marks obtained by the student in computer science practical. Validation restricts this field to hold marks in range 0 to 30.

markse-- It is an integer data member of class student that holds marks obtained by the student in English. Validation restricts this field to hold marks in range 0 to 100.

tot- It is an integer data member that calculates the total marks of a student.

per- It is an integer point data member of class student that holds the percentage of marks obtained by the student. Percentage marks are calculated by dividing total marks by 5.

grade- It is a character data member of class student that holds the grade obtained by student. The system calculates the grade of a student as follows:

PERCENTAGE	**GRADE**
>=90	**A**
>=80 &<90	**B**
>=70&<80	**C**
<70	**F**

Member functions: All the member functions given below are public.

getadno()-It is an accessor function which returns the admission number of a student of a class.

getclass()- It is an accessor function which returns the class of a student of a class.

report()- This member function displays the report card of a student.

calculate()- This is a member function which calculates the grade, total and percentage of a student of a class

tab()- This member function displays the details of all students of a class in tabular form.

display()- This displays the details of an individual student of a class.

input()- this member function takes all values of a student for a report card.

IV. OVERVIEW OF C++

C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms. Bjarne Stroustrup began in work on "C with Classes" in 1979. At first, the class, derived class, strong type checking, in lining, and default argument features were added to C. In 1983, the name of the language was changed from C with Classes to C++. New featured were added including virtual functions, function and operator overloading, references, constants, user controlled free store memory control, improved type checking, and single comment with two forward slashes(//). Release 2.0 of C++ came in 1989 and the updated second edition of the C++ Programming Language was released in 1991. New features included multiple inheritance, abstract classes, static member functions, constant member functions and protected members.

As the C++ language evolved, the standard library evolved with it. The final addition to the C++ standard library was the I/O library which provided facilities to replace the traditional C functions such as printf and scanf. Later, among the most significant additions to the standard library, was large amount of the standard Template Library.

C++ is sometimes called a hybrid language. It is possible to write object oriented or procedural code in the same program in C++. C++ continues to be and is one of the preferred programming languages to develop professional applications.

Here is a brief description about various features provided by C++

Data Abstraction: Data abstraction is a process of representing the essential features without including the background details or implementing details. Abstraction supports data hiding so that only relevant information is exposed to the user and the rest of the information remains hidden from the user. C++ introduces classes that is a vehicle for data abstraction. The class groups its members (data and functions)

into three sections: private, protected and public, where private and protected are hidden from the outside world. Only the public members, that are relevant for the outside world, remain visible.

Encapsulation: Encapsulation refers to data and associated functions into a single unit. Using the method of encapsulation, the programmer cannot directly access the data. Data is only accessible through the functions associated with it. C++ class binds the data and associated member functions into a single unit. The concept of encapsulation shows that a nonmember function cannot access or accidently change an object's private and protected data

Inheritance: Inheritance allows one data type to acquire properties of other data type. It is a mechanism of reusing and extending existing classes without modifying them, thus producing hierarchical relationships between them. Thus, inheritance supports re-usability of the code and is able to simulate the transitive nature of real life objects.

In C++, subclass can derive itself publicly, protectedly, or privately. This visibility mode controls the access specifier to be, for inheritance members of base class. In other words, it determines whether non-member functions and derived classes can access the inherited public and protected members of the base class.

In the publicly derived class, the public and protected members of the base class remain public and protected respectively. In privately derived class, the public and protected members of the base class become private members of the derived class and in the protected protectedly derived class, the public and protected members of the base class become protected members of the derived class. Multiple inheritance is a C++ feature not found in most other languages, allowing a class to be derived from more than 1 base classes, this allows more elaborate inheritance relationships.

Polymorphism: It is key to the power of object oriented programming. Polymorphism is the concept that supports the capability of an object of a class to behave differently in response to messages or action. The primary usage of polymorphism is the ability of objects belonging to different types to respond to method calls of the same name, each one according to an appropriate type-specific behavior. The caller does not have to know the exact type of the callee(called object), thus the exact behavior is determined during run time. In C++, polymorphism is achieved through virtual functions.

Modularity: Modularity id partitioning a program into cohesive and loosely coupled modules. It means that a system can be easily composed of modules, that, when taken out of this specific system, can be combined to produce new systems. Modularity reduces complexity, helps several programmers to work on individual programs at the same time. The code becomes easier to debug, update and modify. It leads to a structured approach as a complex program can be broken into simpler tasks. In C++, classes provide modularity and structure to the program. In addition, modularity can also be achieved through separately compiled files.

Classes: Class represents a group of similar objects. Class is a way to bind data describing an entity and its associated functions together. Classes provide modularity and structure in an object-oriented program. The class is the fundamental OOP concept in C++. Class includes the datamembers, member functions, access specifier and class tag names. Data members are data type properties that describe the characteristics of a class. Member functions describe what and how a class can do. They are set of operations performed for class members. Public member functions provide mechanism to work with objects of a class to the outside world. Through access specifiers, class controls which code has access to its members. There are three main types of access specifiers in C++ private, public and protected.

A private member within a class denotes that only members of the same class have accessibility. The private members are hidden from outside world. They implement the data hiding concept of OOP. Public members are accessible from outside the class. They cannot be accessed by non-members functions. However protected members are inheritable and are accessible in the derived class. Thus C++ class supports OOP feature of encapsulation by binding data and associated functions together. It supports abstraction by providing only the essential information to outside world. It supports data hiding by hiding information from outside world through private and protected members.

Objects: An object is an instantiation of a class. In terms of variables, a class would be the type, and an object would be the variable. Object is implemented by representing its state through functions. Memory space is allocated to the object's data members at the time of their declaration. Member functions are created and placed in memory space only once when the time of their declaration. Member functions are created and placed in memory space only once when the class is defined. Like variables, objects can be passed as function arguments. And functions can also return an object.

Constructor and Destructor: C++ provides special member functions for initializing and disposing of objects belonging to a class. Constructor is a member function of a class with same name as that of a class, which is automatically called when an object is created of that class. The primary job of the constructor is to initialize the object to the legal initial value for the class. If a class has a constructor, each object of the

class will be initialized before any use if made of the object. If a class has no constructor defined, the compiler will supply a default constructor, which allocated memory to data members of the object.

Just as objects are created they have to be destroyed. A destructor is a member function that deallocates the resources allocated to an object before it is destroyed. If there is no explicit destructor the compiler automatically generates default destructor.

Function Overloading: A function name having several definitions that are differentiable by number or type of their arguments is called as function overloading. Overloaded functions have same name but different signatures. Function overloading not only implements polymorphism but also reduces number of comparisons making program faster. The matching process follows the sequence by searching for exact match, a match through type promotions, a match through standard conversion or a match through user defined conversion. This information is known to compiler at the compile time. Hence the compiler chooses the right function at the compile time, which is called as early binding or static binding.

Friend function: A friend function of a class is a non-member of the class that is declared as a friend using the keyword “friend" inside the class. By declaring a function as a friend it gets the right to access all the private and protected members of the class. Even though the prototypes for friend functions appear in the class definition friends are not member functions.

Inline functions: The inline functions are a C++ enhancement designed to speed up programs. The keyword "inline" or placement the function definition inside the class, is a hint to the compiler to make the function inline. With inline functions, the compiler replaces the function call statement with function code and then compiles the entire code. Thus with inline functions, the compiler does not have to jump to another location to execute the function and then jump back.

This pointer: Every object in C++ has access to its own address through an important pointer called this pointer. The "this" pointer is an implicit parameter to all member functions. Therefore inside a member function, this may be used to refer to the invoking object .Friend functions and static functions do not have a this pointer.

Dynamic memory allocation: Then the amount of memory to be allocated is not known beforehand rather it is required to allocate memory as and when required during run time it is called as dynamic memory allocation. C++ offers two operators for dynamic memory allocation- new and delete. The new operator allocates memory dynamically and returns the pointer storing the memory address of the allocated memory. The operator delete de allocates the memory pointed by the given pointer.

V.HEADER FILES

Sl.No	Header File	Functions Used	Description
1.	fstream.h (includes iostream.h)	open()	opens file in specific mode
		close()	closes file associated with the object
		read((char*)&buf, sizeof(buf))	Reads specified number of bytes from associated stream and puts it in buffer pointed buf.
		seekg(long)	sets the position of the get pointer
		tellg()	returns the position of the get pointer
		eof()	returns non-zero(true) and End-Of-File is encountered ;otherwise returns zero(false)
		write((char*)&buf, sizeof(buf))	Writes specified number of bytes to associated stream from buffer pointed by buf.
2.	conio.h	getch()	it gets a character through the keyboard
		textmode()	brings the screen to text mode from graphics mode
		clrscr()	clears screen
3.	stdlib.h	atoi()	converts the string to integer
		exit()	terminates the program normally
4.	string.h	strcpy()	copies one string to another
5.	stdio.h	remove()	deletes file specified in filename
		gets()	gets a line of text through keyboard
6.	iomanip.h	setw()	sets field width
7.	ctype.h	isdigit()	checks if character is a digit and returns non-zero if digit ; zero otherwise
		isalpha()	checks if character is an alphabet and returns non-zero if digit ;zero otherwise
8.	graphics.h	setbkcolour()	sets background colour
		setcolour()	sets the font colour
		settextstyle()	Changes the way in which the text appears
		outtextxy()	Displays the text or string at a specified point on the screen

VI. FILE DESIGN STRUCTURE

File name: “report.dat”

Sl no.	Fields	Description
1.	adno	admission number of the student
2.	name	name of a student
3.	sclass	class of the student
4.	fname	fathers name
5.	mname	mothers name
6.	batch	current year
7.	dob	date of birth
8.	marksp	marks obtained by student in Physics Theory
9.	marksc	marks obtained by student in Chemistry Theory
10.	phyp	marks obtained by student in Physics Practical
11.	chemp	marks obtained by student in Chemistry Practical
12.	marksm	marks obtained by student in Math
13.	csp	marks obtained by student in Computer Sc. Practical
14.	markscs	marks obtained by student in Computer Sc. Theory
15.	markse	marks obtained by student in English
16.	per	percentage marks obtained by student
17.	grade	grade obtained by a student
18.	tot	total marks obtained by a student

VII.PROCEDURE/FUNCTION DESCRIPTION

Class name: student

Sl.No	Function Name	Description
1.	input()	This function creates data for a new student. It prompts the user to enter following details: admission number, name, class, fathers name, mothers name, date of birth and marks of following subjects: Physics Theory, Chemistry Theory, Computer Sc. Theory, Math, English, Physics Practical, Chemistry Practical, and Computer Sc. Theory and validates if marks are within range and contains only digits, validates date of birth. It calls function calculate() which calculates total, percentage, grade
2.	display()	This function displays details of an individual student.
3.	tab()	This function displays details of all students of a class in tabular form.
4.	calculate()	This function calculates total, percentage and grade of a student.
5.	getadno()	This function returns the admission number of the student.
6.	getclass()	This function returns the class of the student.
7.	report()	This function displays the report card of a student.

Other Functions

Sl. No.	Function Name	Description
1.	validint()	This function is used to validate integers.
2.	validname()	This function is used to validate strings.

VIII. SOURCE CODE

```
#include<conio.h>
#include<stdio.h>
#include<iomanip.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#include<graphics.h>
#include<fstream.h>
int n=0;
int validname(char ss[50]) //this function is used to validate names
{ int i;
  int x=0;
for(i=0;ss[i]!=0;i++)
{
if(!isalpha(ss[i]))
{
x=1;
break;
}}
if(x==1)return 1;
return 0;
}

int validint(char ss[50])// This function is used to validate strings
{int j; int z=0;
```

```
for(j=0;ss[j]!='\0';j++)
{
if(!isdigit(ss[j]))
{
z=1;
break;
}}
if(z==1)return 1;
return 0;
}
struct dateofbirth
{int d;
int m;
int y;
};
class student
{int adno;
char name[50];
int sclass;
char fname[50];
char mname[50];
int batch;
dateofbirth dob;
int marksp,marksc,phyp,chemp,marksm,csp,markscs,markse;
float per;
char grade;
float tot;
void calculate() //calculates the grade, total and percentage of a student of a class
```

```
{ tot=marksp+marksc+phyp+chemp+marksm+csp+markscs+markse;
per=(marksp+marksc+phyp+chemp+marksm+csp+markscs+markse)/5.0;
if(per>=90)grade='A';
else if(per>=80)grade='B';
else if(per>=70)grade='C';
else grade='F';
}
public:
void input();
void display()//displays the details of an individual student of a class.
{
cout<<"\n\t\t-----Details of a student-----";
cout<<"\n\t\tAdmission Number              :"<<adno;
cout<<"\n\t\tName                         :"<<name;
cout<<"\n\t\tClass                        :"<<sclass;
cout<<"\n\t\tFather's Name                :"<<fname;
cout<<"\n\t\tMother's Name                :"<<mname;
cout<<"\n\t\tBatch                        :"<<batch;
cout<<"\n\t\tDate of birth                :"<<dob.d<<"/"<<dob.m<<"/"<<dob.y;
cout<<"\n\t\t ----------Marks----------";
cout<<"\n\t\tPhysics theory               :"<<marksp;
cout<<"\n\t\tPhysics practical            :"<<phyp;
cout<<"\n\t\tChemistry theory             :"<<marksc;
cout<<"\n\t\tChemistry practical          :"<<chemp;
cout<<"\n\t\tMaths                        :"<<marksm;
cout<<"\n\t\tEnglish                      :"<<markse;
cout<<"\n\t\tComputer Science theory      :"<<markscs;
cout<<"\n\t\tComputer science practical   :"<<csp;
cout<<"\n\t\tTotal                        :"<<tot;
```

```
cout<<"\n\t\tPercentage                  :"<<setprecision(2)<<per;

cout<<"\n\t\tGrade                       :"<<grade;

}

void tab()//displays the details of  all students of a class in tabular form.

{
cout<<setw(6)<<adno<<setw(10)<<name<<setw(6)<<marksp<<setw(7)<<phyp<<setw(7)<<marksc<<set
w(7)<<chemp<<setw(6)<<marksm<<setw(4)<<markse<<setw(6)<<markscs<<setw(6)<<csp<<setw(5)<<
setprecision(2)<<per<<" "<<grade<<endl;

}

int getadno()//returns the admission number of a student of a class

{

return adno;

}

int getclass()//returns the class of a student of a class.

{

return sclass ;

}

void report()//displays the report card of a student

{

clrscr();

cout<<"\n\t----------STUDENT REPORTCARD----------";

cout<<"\n\tGrade:"<<sclass<<"|Year:"<<batch<<"|Adno:"<<adno<<"|Name:"<<name;

cout<<"\n\tSUBJECT                  Max|Min|Marks Obtained";

cout<<"\n\t-------------------------------------------------- ";

cout<<"\n\tEnglish                  100|33|"<<markse;

cout<<"\n\tMaths                    100|33|"<<marksm;cout<<"\n";

cout<<"\n\tPhysics Theory            70|23|"<<marksp;

cout<<"\n\tPhysics Practical         30|10|"<<phyp;

cout<<"\n\t--------------------------------------------------              ";

cout<<"\n\t   Total                 100|33|"<<marksp+phyp;

cout<<"\n\t--------------------------------------------------              ";
```

```
cout<<"\n\tChemistry Theory           70|23|"<<marksc;
cout<<"\n\tChemistry Practical        30|10"<<chemp;
cout<<"\n\t-------------------------------------------------------              ";
cout<<"\n\t   Total                  100|33|"<<marksc+chemp;
cout<<"\n\t-------------------------------------------------------              ";
cout<<"\n\tComputer Science Theory     70|33|"<<markscs;
cout<<"\n\tCOmputer Science Practical  30|10|"<<csp;
cout<<"\n\t-------------------------------------------------------              ";
cout<<"\n\t   Total                  100|33|"<<markscs+csp;
cout<<"\n\t-------------------------------------------------------              ";
int tot=marksp+phyp+marksc+chemp+markse+marksm+markscs+csp;
cout<<"\n\t-------------------------------------------------------              ";
cout<<"\n\t GRAND TOTAL:"<<tot<<"  Percentage:"<<per<<"  Grade:"<<grade;
cout<<"\n\t-------------------------------------------------------              ";
}
}st;
void student::input()//takes all values of a student for a report card.
{int w=0;
 char ss[50];
l1:
cout<<"\n\t\tEnter admission number    :"; gets(ss);w=validint(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l1;}
 adno=atoi(ss);

l2:
cout<<"\n\t\tEnter name           :";gets(ss);w=validname(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l2;}
strcpy(name,ss);
l3:
```

```
cout<<"\n\t\tEnter Class(enter digits)   :";gets(ss);
w=validint(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l3;}
sclass=atoi(ss);
l4:
cout<<"\n\t\tEnter Father's Name    :";gets(ss);
w=validname(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l4;}strcpy(fname,ss);
l5:
cout<<"\n\t\tEnter Mother's Name    :";gets(ss);
w=validname(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l5;}
strcpy(mname,ss);

l6:
cout<<"\n\t\tEnter Year(Batch)        :";gets(ss);
w=validint(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l6;}
batch=atoi(ss); if(batch<2000||batch>2050)
{cout<<"\n\t\t\tWRONG BATCH ENTERED";goto l6;}
l7:cout<<"\n\t\tDate of Birth:";cout<<"\n\t\tEnter day           :";gets(ss);
w=validint(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l7;}
dob.d=atoi(ss);
cout<<"\n\t\tEnter month           :";gets(ss);
w=validint(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l7;}
dob.m=atoi(ss);
cout<<"\n\t\tEnter year            :";gets(ss);
```

```
w=validint(ss);
if(w==1){cout<<"\n\t\t\t\tERROR";goto l7;}
dob.y=atoi(ss);
if(dob.y<2000||dob.d<0||dob.d>31||dob.m<0||dob.m>12)
 {
  cout<<"\n\t\t\tDATE INVALID";goto l7;
 }
else
   {if(dob.m==2&&(dob.d>28||dob.d<0))
            {cout<<"\n\t\t\tDATE INVALID";goto l7;}

    else if(dob.m==1||dob.m==3||dob.m==5||dob.m==7||dob.m==8||dob.m==10||dob.m==12)
             {if(dob.d>31||dob.d<0)
               { cout<<"\n\t\t\tDATE INVALID";goto l7;}
             }
    else if(dob.m==4||dob.m==6||dob.m==9||dob.m==11)
           {if(dob.d>30)
              {  cout<<"\n\t\t\tDATE INVALID";goto l7;}
             }
   }


cout<<"\n\n\t\tEnter marks\n";
a1:
cout<<"\n\t\tphysics theory(70)          :";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a1;}marksp=atoi(ss);
if(marksp>70||marksp<0)
```

```
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a1;}
a2:
cout<<"\n\t\tphysics practical(30)       :";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a2;}phyp=atoi(ss);
if(phyp>30||phyp<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a2;}
a3:
cout<<"\n\t\tchemistry theory(70)        :";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a3;}marksc=atoi(ss);
if(marksc>70||marksc<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a3;}
a4:
cout<<"\n\t\tchemistry practical(30)     :";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a4;} chemp=atoi(ss);
if(chemp>30||chemp<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a4;}
a5:
cout<<"\n\t\tmaths(100)                 :";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\tERROR";goto a5;} marksm=atoi(ss);
if(marksm>100||marksm<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a5;}
```

```
a6:
cout<<"\n\t\tenglish(100)                :";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a6;} markse=atoi(ss);
if(markse>100||markse<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a6;}
a7:
cout<<"\n\t\tcomputer theory(70)          : ";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a7;}markscs=atoi(ss);
if(markscs>70||markscs<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a7;}
a8:
cout<<"\n\t\tcomputer practical(30)       : ";gets(ss);
w=validint(ss);
if(w==1)
{cout<<"\n\t\t\t\tERROR";goto a8;}csp=atoi(ss);
if(csp>30||csp<0)
{cout<<"\n\t\t\tWRONG MARKS ENTERED";goto a8;}
calculate() ;
}
void main()
{
  clrscr();
   int gd=DETECT,gm;
  initgraph(&gd,&gm,"C:\\TC\\BGI\\");
  settextstyle(1,0,4);
```

```
 setbkcolor(MAGENTA);
 setcolor(WHITE);
  ofstream file8("report.dat", ios::binary|ios::out) ;
            if(!file8){cout<<"\n\t\tERROR";getch();exit(1);}
            file8.close();      getch();
 outtextxy(120,210,"STUDENT REPORT CARD");getch();
 L1: clrscr();
  textmode(3);
  setbkcolor(BLUE);
char ch[50];



 cout<<"\n\n\n\n\n\t\t----------REPORTCARD GENERATION----------\n";
 cout<<"\n\t\t1.Create a student record";
 cout<<"\n\t\t2.Display a student record";
 cout<<"\n\t\t3.Display student details of a class";
 cout<<"\n\t\t4.Modify a student record";
 cout<<"\n\t\t5.Delete a student record";
 cout<<"\n\t\t6.Display reportcard of a student";
 cout<<"\n\t\t7.Exit";
 b1:cout<<"\n\n\t\tEnter choice   :";
 gets(ch);
int w=validint(ch);
 if(w==1){cout<<setw(20)<<"\n\t\t\tERROR";goto b1;}int c=atoi(ch);
 switch(c)
 { case 1:clrscr(); char ch4;
            ofstream f1("report.dat",ios::binary|ios::app);
            if(!f1){cout<<"error";exit(1);}
```

```
        cout<<"\n\t\t----------STUDENT REPORTCARD GENERATION---------";


        st.input();n++;
        f1.write((char *)&st,sizeof(st));
        cout<<"\n\t\t\tSTUDENT RECORD SUCCESSFULLY ADDED";


        f1.close();getch(); goto L1;
case 2://display details of a student
        clrscr();char ss[50];if(n!=0)
        {
         cout<<"\n\t\t----------STUDENT REPORTCARD GENERATION----------\n";int adno;
         L2:ifstream f2("report.dat",ios::binary);
         if(!f2){cout<<"Error";exit(1);}
         cout<<"\n\t\tEnter admission number    :";
         gets(ss);
         int w1=validint(ss);
          if(w1==1)
                {cout<<"\n\t\tINVALID INPUT";
                 goto L2;
                }
         adno=atoi(ss); int f7=0;
         while(!f2.eof())
          {f2.read((char *)& st,sizeof(st));
          if(f2.eof())break;
          if(adno==st.getadno())
            {st.display();getch();f7++;}
          }
```

```
if(f7==0)
 {cout<<"\n\t\t\tADMISSION NO. NOT FOUND";getch();} }
  else {gotoxy(25,10);cout<<"STUDENT DETAILS NOT ENTERED";getch();
} goto L1;
case 3://display student details of a class
clrscr(); char sc[50]; ifstream f3("report.dat",ios::binary);
if(!f3){cout<<"Error";exit(1);}
if(n!=0){
cout<<"\n\t\t----------STUDENT REPORTCARD GENERATION----------\n";int clas;
L3:cout<<"\n\t\tEnter the class to be displayed  :";
gets(sc);cout<<"\n";
int w2=validint(sc);
if(w2==1)
{cout<<"\n\t\t\t\t ERROR";goto L3;}clas=atoi(sc);int f=0;
 while(!f3.eof())
 {f3.read((char *)& st,sizeof(st));
  if(f3.eof())break;
  if(clas==st.getclass())
  {++f;
   if(f==1)

{cout<<setw(6)<<"Ad.No"<<setw(10)<<"Name"<<setw(6)<<"Phy_T"<<setw(7)<<"Phys_P"<<setw(7)<
<"Chem_T"<<setw(7)<<"Chem_P"<<setw(6)<<"Maths"<<setw(4)<<"Eng"<<setw(6)<<"Com_T"<<set
w(6)<<"Com_P"<<setw(5)<<"Per%"<<" "<<"Grade";
     cout<<"\n--------------------------------------------------------------------------------\n";
   }
  st.tab();  getch();
 }
}
if(f==0)
{cout<<"\n\t\t\t NO STUDENT FOUND";getch();
```

```
}}else {gotoxy(25,10);cout<<"STUDENT DETAILS NOT ENTERED";getch();
  }

goto L1;
case 4://to modify student detail
fstream f4("report.dat",ios::binary|ios::in|ios::out);
clrscr();
if(!f4){cout<<"Error";exit(1);}
char sp[50];
if(n!=0)
cout<<"\n\t\t----------STUDENT REPORTCARD GENERATION----------\n";if(n!=0){
L4:cout<<"\n\t\tEnter student admission no to be modified  :";int f1=0;
gets(sp);
int w3=validint(sp);
if(w3==1)
{cout<<"\n\t\t\t\tERROR";goto L4;} int adno=atoi(sp);
while(!f4.eof())
{int p=f4.tellg();
f4.read((char *)& st,sizeof(st));
 if(f4.eof())break;
if(adno==st.getadno())
{++f1;
 f4.seekp(p);
 cout<<"\n\n\t\t Enter data to be modified  :";
 st.input();
 f4.write((char *)&st,sizeof(st));
 cout<<"\n\t\t\t\tDATA UPDATED";  getch();
}
 }f4.close();
```

```
        if(f1==0)
        {cout<<"\n\t\t\t\tSTUDENT NOT FOUND";getch();}}else {gotoxy(25,10);cout<<"STUDENT
DETAILS NOT ENTERED";getch();
         }
        goto L1;
    case 5://deletion of record
        clrscr();char sl[50];if(n!=0){
        cout<<"\n\t\t-----STUDENT REPORTCARD GENERATION------";
        ifstream t2("report.dat",ios::binary);
        ofstream t3("tempo.dat",ios::binary);
        if(!t2){cout<<"error";exit(0);}
        if(!t3){cout<<"error";exit(0);}
        L52:cout<<"\n\t\tEnter admission number    :";
        gets(sl);
        int w0=validint(sl);
         if(w0==1)
              {cout<<"\n\t\tINVALID INPUT";
               goto L52;
              }
        int adno=atoi(sl); int f=0;
        while(!t2.eof())
        {t2.read((char*)&st,sizeof(st));
         if(t2.eof())break;
         if(st.getadno()!=adno)
         { t3.write((char*)&st,sizeof(st));}
           else
               f++;
        }t2.close();
        t3.close();
        remove("report.dat");rename("tempo.dat","report.dat");
```

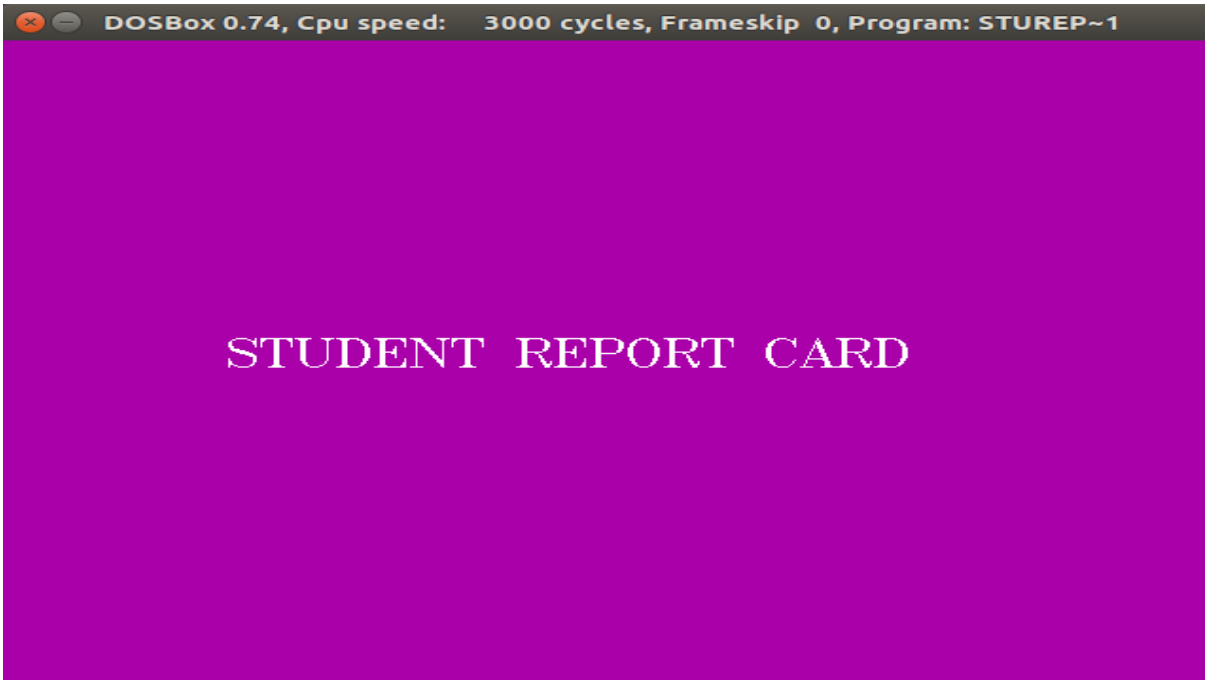
```
        if(f==0)
         {cout<<"\n\t\t\t\tRECORD NOT FOUND"; getch();}
          else
        { cout<<"\n\t\t\tRECORD IS DELETED"; getch();}
         }
        else {gotoxy(25,10);cout<<"STUDENT DETAILS NOT ENTERED"; getch();}
goto L1;
case 6://display reportcard of a student
        clrscr(); char sd[50]; if(n!=0){
        cout<<"\n\t\t----------STUDENT REPORTCARD GENERATION----------\n";
        L5:cout<<"\n\t\tEnter admission number  :";
        gets(sd);
        int w8=validint(sd);int f=0;
        if(w8==1)
        {cout<<"\n\t\t\t\tERROR";goto L5;} int adno=atoi(sd);
        ifstream t1("report.dat",ios::binary);
        if(!t1){cout<<"error";exit(0);}
        while(!t1.eof())
        {t1.read((char*)&st,sizeof(st));
         if(t1.eof())break;
          if(adno==st.getadno())
            {++f;
                st.report();getch();

            }
        }  t1.close();
        if(f==0)
         {cout<<"\n\t\t\t\tRECORD NOT FOUND";getch();}}
```

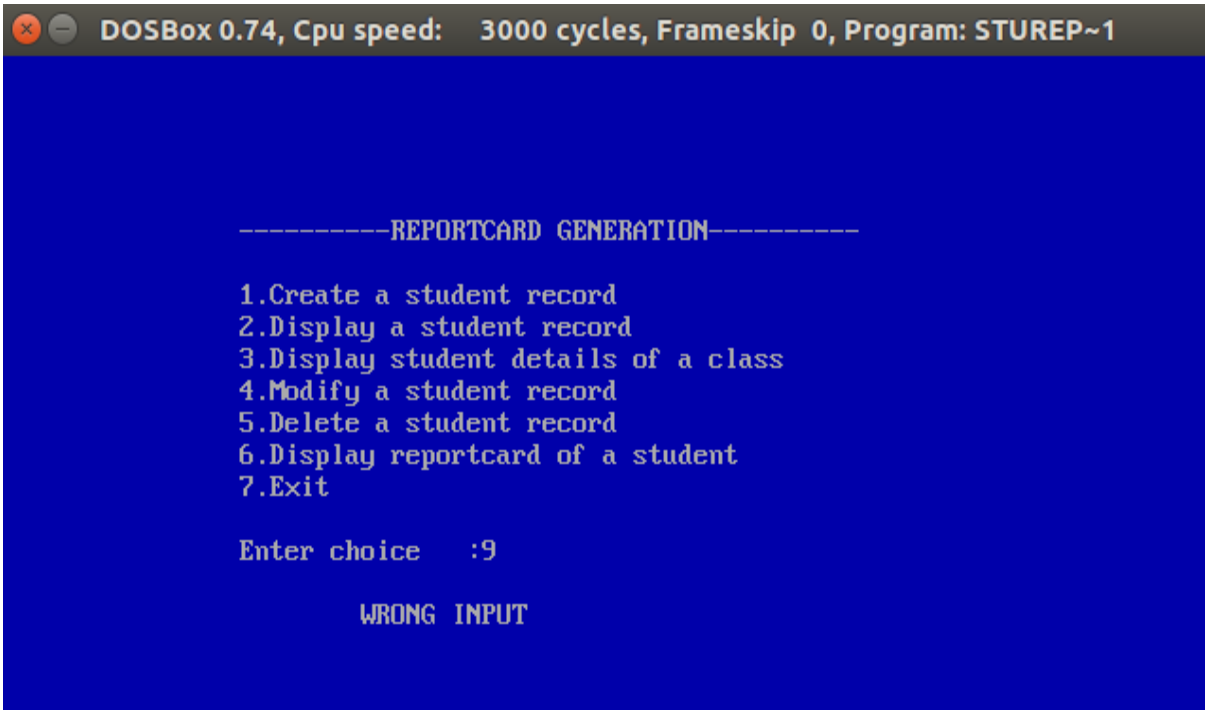
```
          else {gotoxy(25,10);cout<<"STUDENT DETAILS NOT ENTERED"; getch();}
          goto L1;
  case 7:/* ofstream file8("report.dat", ios::binary|ios::out) ;
          if(!file8){cout<<"\n\t\tERROR";getch();exit(1);}
          file8.close();  n=0;*/
          exit(0);    goto L1;
  default:cout<<"\n\t\t\tWRONG INPUT";getch();  goto L1;
}
//  goto L1;
  }
```

IX.OUTPUT SCREENS

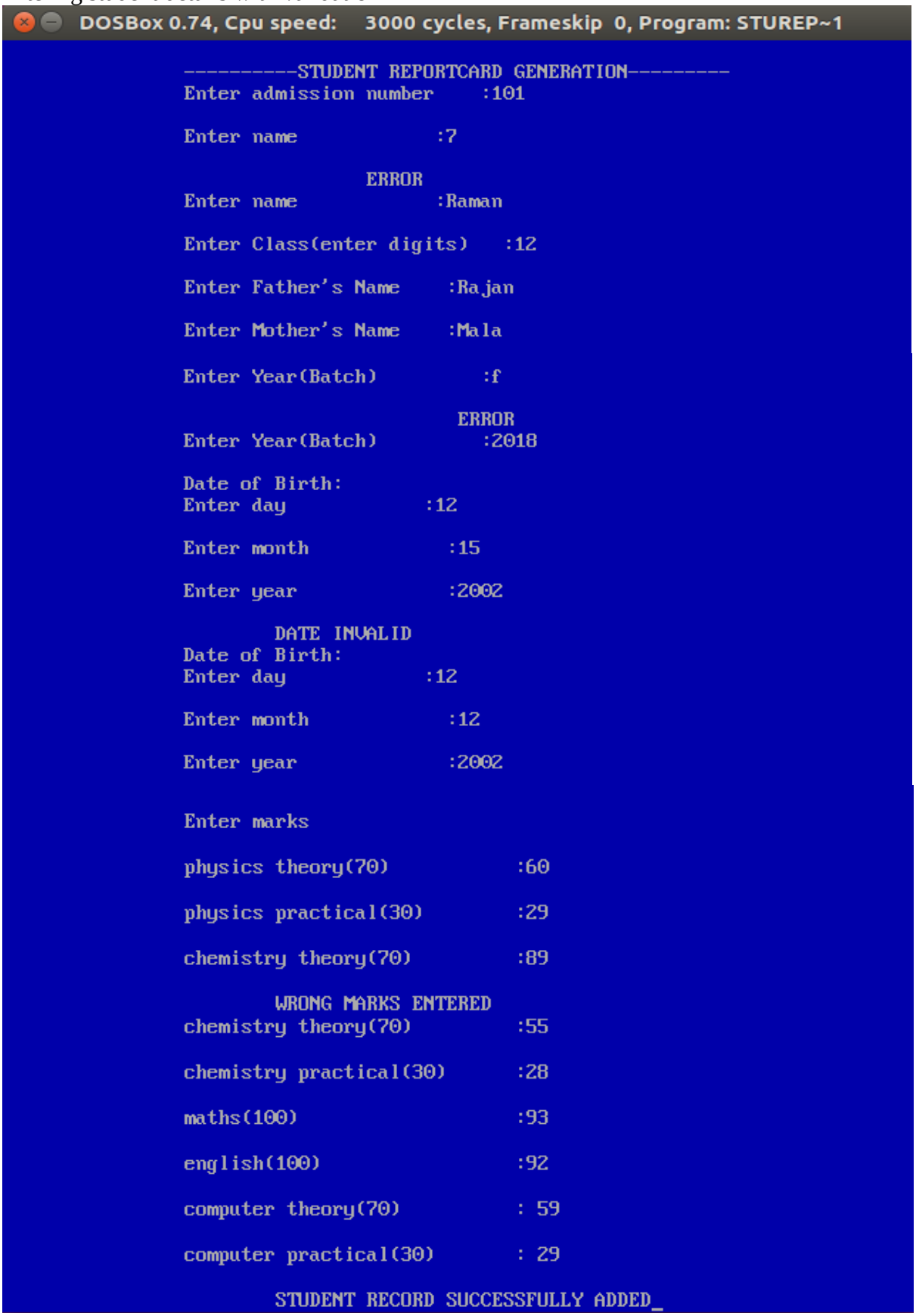
Title Screen



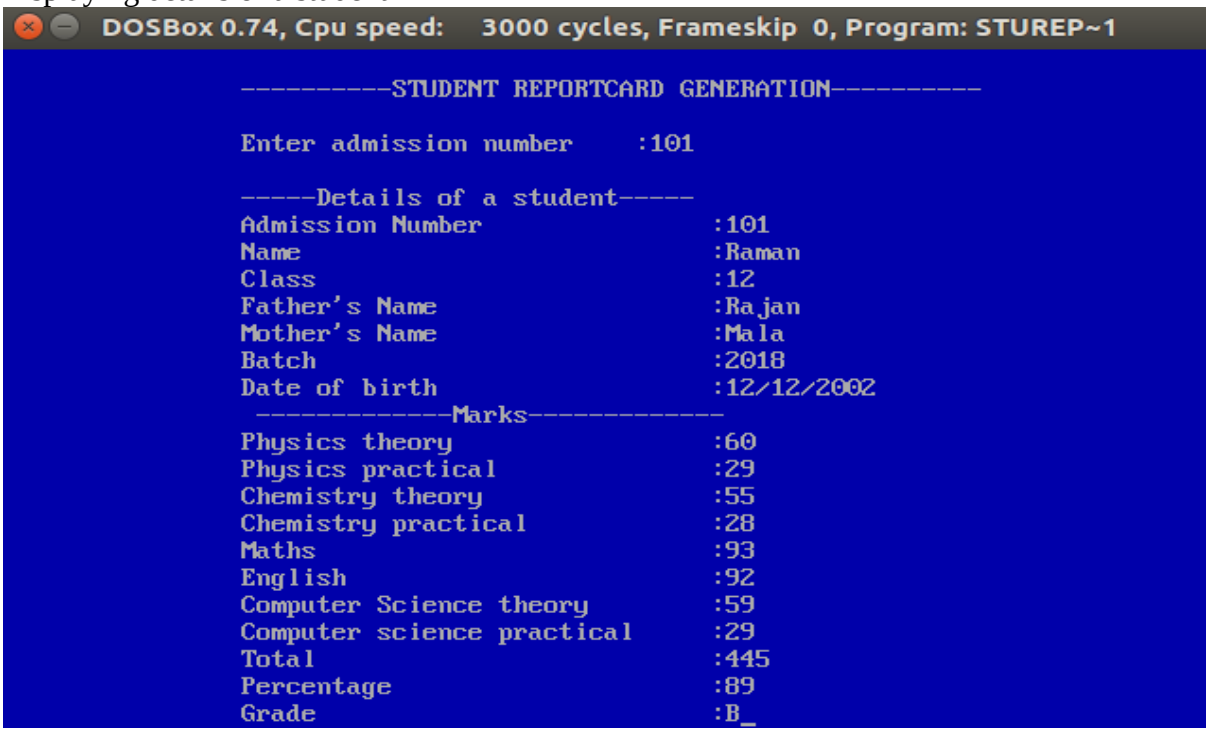
Main menu with validation



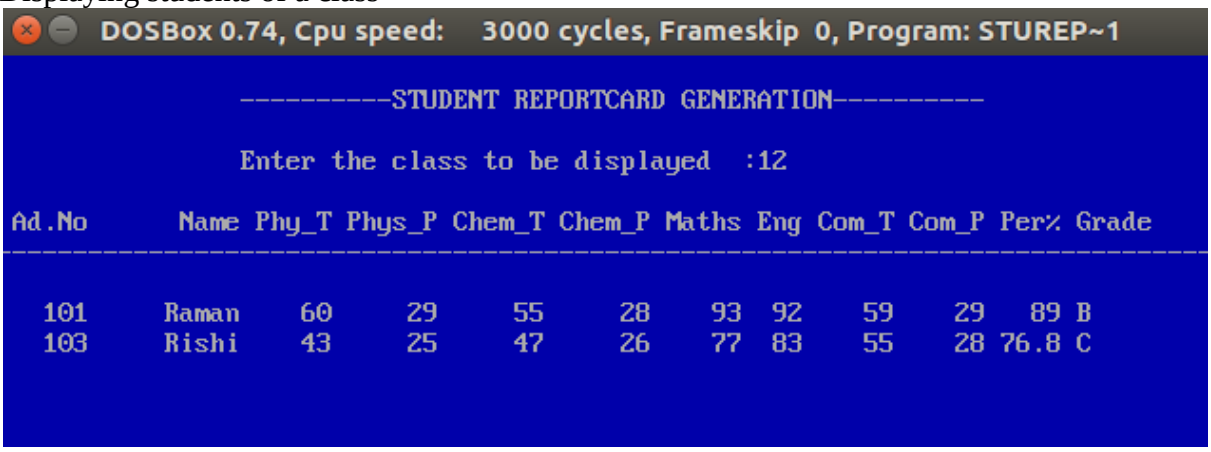
Entering student details with validation



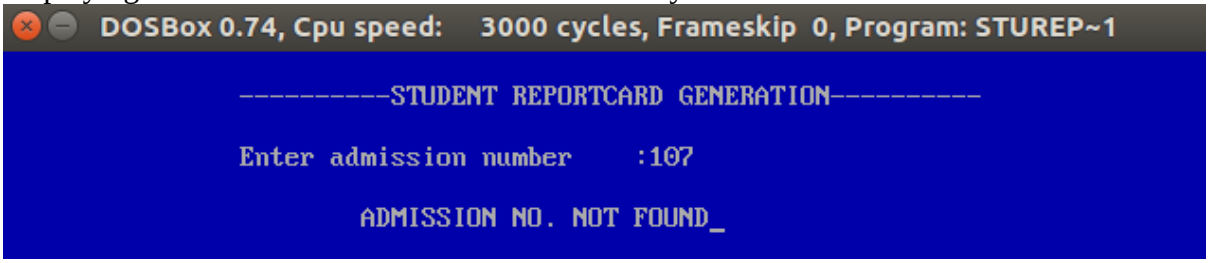
Displaying details of a student



Displaying students of a class



Displaying details of a student with invalid data entry



Modifying student's details