Ex. No. 6	**Collections**		
Date of Exercise	25.08.2016	Date of Upload	23.10.2016

Aim

To develop **Student Information System** using C# by implementing Lists for storing and retrieving values.

Description

Collection classes are specialized classes for data storage and retrieval. These classes provide support for stacks, queues, lists, and hash tables. Most collection classes implement the same interfaces.

Collection classes serve various purposes, such as allocating memory dynamically to elements and accessing a list of items on the basis of an index etc. These classes create collections of objects of the Object class, which is the base class for all data types in C#.

Various Collection Classes and Their Usage

The following are the various commonly used classes of the **System.Collection** namespace.

Class	**Description and Usage**
ArrayList	It represents ordered collection of an object that can be **indexed** individually. It is basically an alternative to an array. However, unlike array you can add and remove items from a list at a specified position using an **index** and the array resizes itself automatically. It also allows dynamic memory allocation, adding, searching and sorting items in the list.
Hashtable	It uses a **key** to access the elements in the collection.

	A hash table is used when you need to access elements by using key, and you can identify a useful key value. Each item in the hash table has a **key/value** pair. The key is used to access the items in the collection.
SortedList	It uses a **key** as well as an **index** to access the items in a list. A sorted list is a combination of an array and a hash table. It contains a list of items that can be accessed using a key or an index. If you access items using an index, it is an ArrayList, and if you access items using a key, it is a Hashtable. The collection of items is always sorted by the key value.
Stack	It represents a **last-in, first out** collection of object. It is used when you need a last-in, first-out access of items. When you add an item in the list, it is called **pushing** the item and when you remove it, it is called **popping** the item.
Queue	It represents a **first-in, first out** collection of object. It is used when you need a first-in, first-out access of items. When you add an item in the list, it is called **enqueue** and when you remove an item, it is called **deque**.
BitArray	It represents an array of the **binary representation** using the values 1 and 0. It is used when you need to store the bits but do not know the number of bits in advance. You can access items from the BitArray collection by using an **integer index**, which starts from zero.

- **Collection Initializers**

```
ArrayList objectList = new ArrayList() {1, 2}; //non-generic
List < int > intList = new List < int > () {1, 2}; //generic
```

- **Adding Elements**

```
objectList.Add("object1"); objectList.Add("object2"); //non- generic
intList.Add(1); intList.Add(2); //generic
```

AddRange() method of the List < T > class, add multiple elements to the collection at once

- **Inserting Elements**

```
objectList.Insert (2,"object3");
```

If the index set is larger than the number of elements in the collection, ArgumentOutOfRangeException is thrown.

InsertRange() offers the capability to insert a number of elements

- **Accessing Elements**

It access the elements by using an indexer and passing the item number.

e.g. The first item can be accessed with an index value 0

```
ArrayList a1 = objectList[2];
for (int i = 0; i < objectList.Count; i++) //count property
{
Console.WriteLine(objectList [i]);
}
foreach (ArrayList a in objectList)
{
Console.WriteLine(a);
}
```

- **Removing Elements**

```
objectList.RemoveAt(3);
objectList.Remove("object1");
```

RemoveAt() method remove the item in the given index, whereas Remove() method first searches in the collection to get the index of the item with the IndexOf() method, and then uses the index to remove the item. RemoveRange() removes a number of items from the collection where the first parameter specifies the index where the removal of items should begin and the second parameter specifies the number of items to be removed.

```
int index = 3; int count = 5;
objectList.RemoveRange(index, count);
```

- **Searching**

Various methods used for searching are IndexOf(), LastIndexOf(), FindIndex(), FindLastIndex(), Find(), and FindLast().

IndexOf() requires an object as parameter and returns the index of the item if it is found and return – 1 if the item is not found. It uses the IEquatable interface for comparing the elements

```
int index1 = objectList.IndexOf(object1);
```

FindIndex(), Find(), FindAll(), FindLast() requires a parameter of type Predicate :

```
int index3 = racers.FindIndex(r = > r.Country == "Finland");
```

- **Type Conversion**

List<T> method ConvertAll() , all types of a collection can be converted to a different type

e.g.

```
List < Person > persons = racers.ConvertAll < Person > (
r = > new Person(r.FirstName + " " +r.LastName));
```

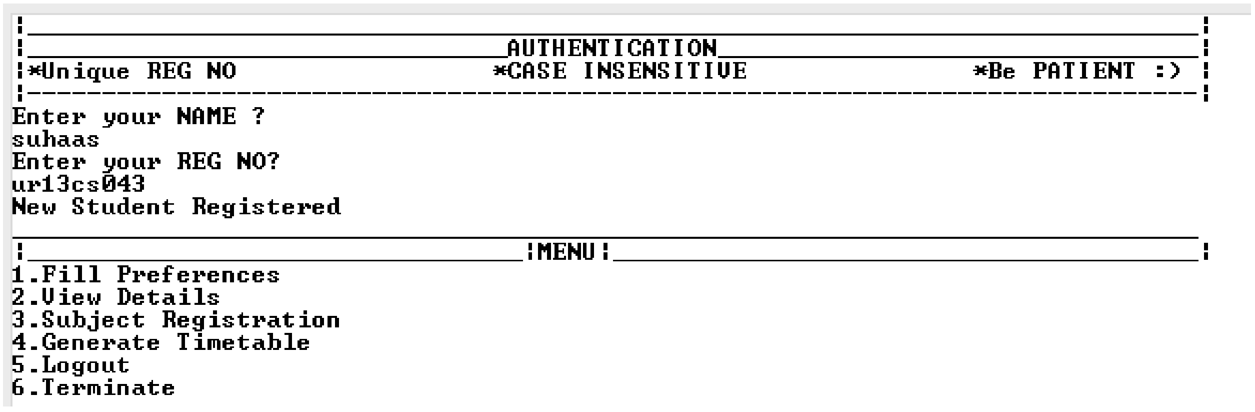
Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Student_Information_System_Collections
{
    class SubjectRegistration
    {
        public string SubjectCode;
        public int SubjectCapacity;
        public string SubjectCategory;
        public int SubjectCredit;
        public string SubjectName;
        public string SubjectStatus;
        public string GroupCode;
    }

    class RegisteredSubject
    {
        public string SubjectCode;
        public string StudentId;
    }

    class Students_Details {
        public string Student_Name;
        public string Student_Regno;
        public string Student_Branch;
        public string Student_Current_Sem;
        public string Student_Mailid;
        public string Student_Mobileno;
        public string Student_Mentor;
    }

    class Registered_Students
    {
        public string Reg_Student_Name;
        public string Reg_Student_Regno;
    }

    class Program
    {
        public static List<SubjectRegistration> subjects = new
List<SubjectRegistration>();
        public static List<Registered_Students> studentreg = new
List<Registered_Students>();
        public static List<Students_Details> studetails = new List<Students_Details>();
        public static List<RegisteredSubject> regsubjects = new
List<RegisteredSubject>();

        static void Main(string[] args)
        {
            initialisevalue(subjects);
            Student_Login(studentreg,studetails);
        }

        public static int initialisestudent(List<Registered_Students> student)
        {
            Console.Clear();

Console.WriteLine("|_____________________________________________________________________
_________|");

Console.WriteLine("|________________________________AUTHENTICATION_______________________
_________|");
            Console.WriteLine("|*Unique REG NO\t\t\t*CASE INSENSITIVE\t\t*Be PATIENT :)
|");
            Console.WriteLine("|---------------------------------------------------------
---------------------|");
            Console.WriteLine("Enter your NAME ?");
            string Name = Console.ReadLine().ToUpper();
            Console.WriteLine("Enter your REG NO?");
            string Id = Console.ReadLine().ToUpper();
            int index = -1;
            index = student.FindIndex(a => a.Reg_Student_Regno == Id);
            if (index < 0)
            {
                Console.WriteLine("New Student Registered");
                student.Add(new Registered_Students() { Reg_Student_Name = Name,
Reg_Student_Regno = Id });
            }
            index = student.FindIndex(a => a.Reg_Student_Regno == Id);
            return index;

        }

        public static void Student_Panel(List<Registered_Students> studentreg,
List<Students_Details> studetails,int index) {

Console.WriteLine("______________________________________________________________________
_________");

Console.WriteLine("|_________________________________|MENU|______________________________
_________|");
            Console.WriteLine("1.Fill Preferences\n2.View Details\n3.Subject
Registration\n4.Generate Timetable\n5.Logout\n6.Terminate");
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice)
            {
                case 1:
                    Student_Fill_Details(studentreg, studetails,index);
                    break;
                case 5:
                    Student_Login(studentreg, studetails);
                    break;
                case 6:
                    Environment.Exit(0);
                    break;
                case 2:
                    Student_Display_Details(studentreg, studetails, index);
                    break;
                case 3:initialisemenu(subjects,index,studentreg,regsubjects);
                    break;
                case 4:
                    generatetimetable(subjects, index, studentreg, regsubjects);
                    break;
                default:
                    break;
            }


        }

        public static void Student_Display_Details(List<Registered_Students> studentreg,
List<Students_Details> studetails, int index)
        {
            string regno = studentreg[index].Reg_Student_Regno, name =
studentreg[index].Reg_Student_Name;
            int detail_index = -1;
            detail_index = studetails.FindIndex(a => a.Student_Regno.Equals(regno));
            if (detail_index == -1)
            {
                Console.WriteLine("The details are stil Not filled!");
                Console.WriteLine("Plese fill out the details");
                Student_Fill_Details(studentreg, studetails, index);
            }
            else {

Console.WriteLine("|____________________________________________________________________
_________|");
                Console.WriteLine("|___________________________STUDENT
INFORMATION________________________________|");
                Console.WriteLine("|Student NAME:\t\t" + name );
                Console.WriteLine("|Student REG NO:\t\t" + regno);
                Console.WriteLine("|Mentor
Name:\t\t"+studetails[detail_index].Student_Mentor);
                Console.WriteLine("|----------------------------------------------------
-------------------------|");
                Console.WriteLine("|\t\t\tAcademics");
                Console.WriteLine("|Branch:\t" + studetails[detail_index].Student_Branch
+ "\t\tCurrent Sem:\t" + studetails[detail_index].Student_Current_Sem);
                Console.WriteLine("|----------------------------------------------------
-------------------------|");
                Console.WriteLine("|\t\t\tContact Details");
                Console.WriteLine("|Phone Number:\t" +
studetails[detail_index].Student_Mobileno+"\t\tMail Id:\t"+
studetails[detail_index].Student_Mailid);

Console.WriteLine("|____________________________________________________________________
_________|");
            }
            Student_Panel(studentreg, studetails, index);
        }

        public static void Student_Fill_Details(List<Registered_Students> studentreg,
List<Students_Details> studetails, int index)
        {
            string name = studentreg[index].Reg_Student_Name, regno =
studentreg[index].Reg_Student_Regno;

            Console.WriteLine("NAME:\t" + name + "\n" + "REG NO:\t" + regno);
            int detail_index = 0;
            detail_index = studetails.FindIndex(a => a.Student_Regno.Equals(regno));
            if (detail_index == -1)
            {
                Console.WriteLine("Enter Your Branch Name ?");
                String branch_name = Console.ReadLine().ToUpper();
                Console.WriteLine("Enter Your Current Sem ?");
                String current_sem = Console.ReadLine().ToUpper();
                Console.WriteLine("Enter Your Mail ID  ?");
                String mail_id = Console.ReadLine();
                Console.WriteLine("Enter Your Phone Number ?");
                String phone_num = Console.ReadLine().ToUpper();
                Console.WriteLine("Enter Your Mentor Name ?");
                String mentor = Console.ReadLine().ToUpper();
                studetails.Add(new Students_Details() { Student_Name = name,
Student_Regno = regno, Student_Branch = branch_name, Student_Current_Sem = current_sem,
Student_Mailid = mail_id, Student_Mentor = mentor, Student_Mobileno = phone_num });
                Console.WriteLine("Details are added successfully");
                Student_Panel(studentreg, studetails, index);
            }
            else
            {
                Console.WriteLine("Dear "+name+",you have filled your details already");
                Student_Display_Details(studentreg, studetails, index);
            }

            Student_Panel(studentreg, studetails, index);
        }

        public static void Student_Login(List<Registered_Students> studentreg,
List<Students_Details> studetails) {
            int index;
            index = initialisestudent(studentreg);
            Student_Panel(studentreg,studetails,index);
        }

        public static void registersubject(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {
            displaysubject(subjects);
            Console.WriteLine("Enter the Subject Code to Register or (N) to stop");
            string Subcode = Console.ReadLine();
            if (Subcode.Equals("N")) { subjectregistration(subjects, studentindex,
student, regsubjects); };
            Subcode = Subcode.ToUpper();

            string sid = student[studentindex].Reg_Student_Regno;
            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode == Subcode &&
s.StudentId == sid);
            if (checkval == -1)
            {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].Reg_Student_Name);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity--;
                        Console.WriteLine("You are successfully registered for " +
subjects[index].SubjectName);
                        regsubjects.Add(new RegisteredSubject() { SubjectCode = Subcode,
StudentId = sid });
                        registersubject(subjects, studentindex, student, regsubjects);

                    }
                    else
                    {
                        Console.WriteLine("No seats are further available :( ");
                        Console.WriteLine("Please try again :) ");
                        subjectregistration(subjects, studentindex, student,
regsubjects);
                    }
                }
                else
                {
                    Console.WriteLine("Sorry!! :( The subcode is not found");
                    Console.WriteLine("Please try again :) ");
                    subjectregistration(subjects, studentindex, student, regsubjects);
                }


            }
            else
            {
                Console.WriteLine("Already registered");
                Console.WriteLine("Please try again :) ");
                subjectregistration(subjects, studentindex, student, regsubjects);
            }



        }

        public static void deregistersubject(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {

            string sid = student[studentindex].Reg_Student_Regno;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            Console.WriteLine("You have registered for the following subjects");

            foreach (RegisteredSubject subval in subs)
            {
                String subcodeval = subval.SubjectCode;
                List<SubjectRegistration> subject1 = subjects.FindAll(a => a.SubjectCode
== subcodeval);
                displaysubject(subject1);
            }

            Console.WriteLine("Enter the Subject Code to Deregister");
            string Subcode = Console.ReadLine();
            Subcode = Subcode.ToUpper();

            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode == Subcode &&
s.StudentId == sid);
            if (checkval != -1)
            {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].Reg_Student_Name);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity++;
                        Console.WriteLine("You are successfully \"DEREGISTERED\" for " +
subjects[index].SubjectName);
                        regsubjects.RemoveAll(a => a.SubjectCode == Subcode &&
a.StudentId == sid);
                        initialisemenu(subjects, studentindex, student, regsubjects);
                    }
                }
                else
                {
                    Console.WriteLine("Sorry!! :( The subject code is not found");
                    Console.WriteLine("Please try again :) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                }


            }
            else { Console.WriteLine("Not Found"); }

        }

        private static void initialisemenu(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {
            subjectregistration(subjects, studentindex, student, regsubjects);
        }

        public static void generatetimetable(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {

            string sid = student[studentindex].Reg_Student_Regno;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            if (subs.Count != 0)
            {

Console.WriteLine("|_________________________________________________________________
_________|");
```

```
                Console.WriteLine("|_________________________________|TIME
TABLE|_________________________________|");

Console.WriteLine("Hour|||\tMonday|||\tTuesday|||\tWednesday|||\tThursday|||\tFriday|||")
;
                foreach (RegisteredSubject subval in subs)
                {
                    String subcodeval = subval.SubjectCode;
                    int indexvalof = subjects.FindIndex(a => a.SubjectCode ==
subcodeval);
                    String groupval = subjects[indexvalof].GroupCode;
                    //Console.WriteLine("Group:" + groupval);
                    if (groupval.Equals("A"))
                    {
                        Console.WriteLine();
                        Console.Write("1\t");
                        for (int i = 0; i < 4; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }
                        Console.WriteLine();
                    }
                    else if (groupval.Equals("B"))
                    {
                        Console.WriteLine();
                        Console.Write("2\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else if (groupval.Equals("C"))
                    {
                        Console.WriteLine();
                        Console.Write("3\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else if (groupval.Equals("D"))
                    {
                        Console.WriteLine();
                        Console.Write("4\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else if (groupval.Equals("E"))
                    {
                        Console.WriteLine();
                        Console.Write("5\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else
                    {
                        Console.WriteLine("_____________You are free___________");
                    }

                }
                Student_Panel(studentreg, studetails, studentindex);
            }
            else {
                Console.WriteLine("No subjects to Display :( ");
                Console.WriteLine("You have still Not registered the subjects :( ");
                initialisemenu(subjects, studentindex, student, regsubjects);
            }
        }

        public static void subjectregistration(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects) {

Console.WriteLine("______________________________________________________________________
_________");
            Console.WriteLine("|_________________________|SUBJECT
REGISTRATION|_______________________________|");
            Console.WriteLine("1.Register Subject");
            Console.WriteLine("2.Deregister Subject");
            Console.WriteLine("3.Faculty Selection");
            Console.WriteLine("4.Generate Timetable");
            Console.WriteLine("5.Go Back To Main Menu");
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice)
            {
                case 1:
                    registersubject(subjects, studentindex, student, regsubjects);
                    break;
                case 2:
                    deregistersubject(subjects, studentindex, student, regsubjects);
                    break;
                case 3:
                    Console.WriteLine("The function is still under Construction ;) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                    break;
                case 4:
                    generatetimetable(subjects, studentindex, student, regsubjects);

                    break;
                case 5:
                    Student_Login(studentreg, studetails);
                    break;
                default:
                    Console.WriteLine("Invalid Choice");
                    break;
            }
        }

        public static void initialisevalue(List<SubjectRegistration> subjects)
        {
            subjects.Add(new SubjectRegistration() { SubjectName = "C# PROGRM",
SubjectCode = "14CS2054", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 4, SubjectStatus = "Available", GroupCode = "A" });
            subjects.Add(new SubjectRegistration() { SubjectName = "STORAGEAN",
SubjectCode = "14CS2065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "B" });
            subjects.Add(new SubjectRegistration() { SubjectName = "BIG _DATA",
SubjectCode = "14CS3065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "C" });
            subjects.Add(new SubjectRegistration() { SubjectName = "INTEROFTH",
SubjectCode = "16EC2002", SubjectCategory = "FREEELEC", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "D" });
            subjects.Add(new SubjectRegistration() { SubjectName = "DATASTRUC",
SubjectCode = "14CS2009", SubjectCategory = "____CORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "E" });
        }

        public static void displaysubject(List<SubjectRegistration> subjects)
        {

Console.WriteLine("\n__________________________________________________________________
______________");
            Console.WriteLine("SUB_CODE\tNAME\t\tCREDIT\tCATEGORY\tCAPACITY\tSTATUS");
            Console.WriteLine("-------------------------------------------------------------
------------------------");
            foreach (SubjectRegistration subval in subjects)
            {
                Console.WriteLine("{0}\t{1}  \t[{2}]\t{3}\t{4}\t\t{5}",
subval.SubjectCode, subval.SubjectName, subval.SubjectCredit, subval.SubjectCategory,
subval.SubjectCapacity, subval.SubjectStatus);

            }
        }
    }


}
```

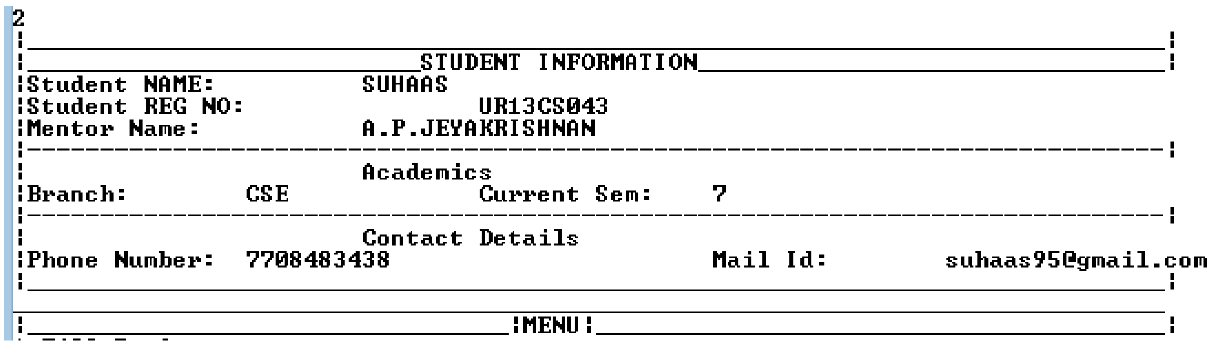
Output

Authentication



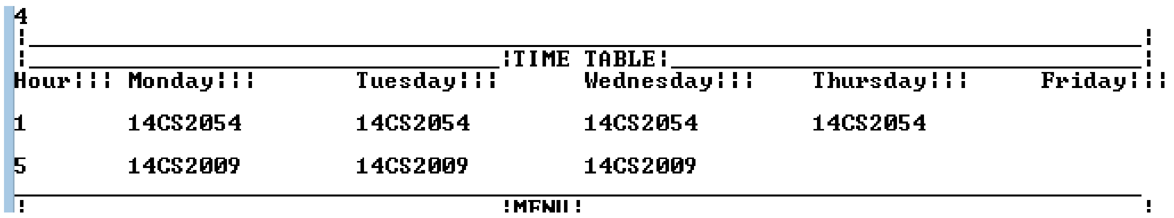
```
¦_____________________________________________________________________________¦
¦________________________________AUTHENTICATION_______________________________¦
¦*Unique REG NO                  *CASE INSENSITIVE                *Be PATIENT :) ¦
¦-----------------------------------------------------------------------------¦
Enter your NAME ?
suhaas
Enter your REG NO?
ur13cs043
New Student Registered
_______________________________________________________________________________
¦_________________________________¦MENU¦______________________________________¦
1.Fill Preferences
2.View Details
3.Subject Registration
4.Generate Timetable
5.Logout
6.Terminate
```

Student Information



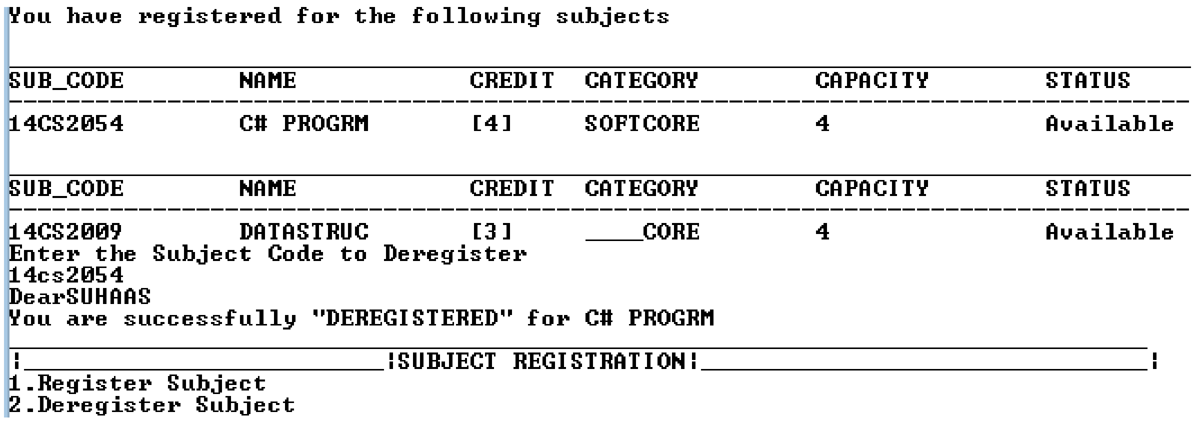
```
2
¦___________________________________________________________________________¦
¦__________________________STUDENT INFORMATION______________________________¦
¦Student NAME:          SUHAAS
¦Student REG NO:                UR13CS043
¦Mentor Name:           A.P.JEYAKRISHNAN
¦---------------------------------------------------------------------------¦
¦                       Academics
¦Branch:        CSE             Current Sem:    7
¦---------------------------------------------------------------------------¦
¦                       Contact Details
¦Phone Number:  7708483438                      Mail Id:        suhaas95@gmail.com
¦___________________________________________________________________________¦
_____________________________________________________________________________
¦________________________________¦MENU¦_____________________________________¦
```

Time table



```
4
¦__________________________________________________________________________¦
¦_______________________________¦TIME TABLE¦_______________________________¦
Hour¦¦¦ Monday¦¦¦       Tuesday¦¦¦      Wednesday¦¦¦      Thursday¦¦¦      Friday¦¦¦

1       14CS2054        14CS2054        14CS2054        14CS2054

5       14CS2009        14CS2009        14CS2009
____________________________________________________________________________
¦                               ¦MENU¦                                     ¦
```

Derigisteration



```
You have registered for the following subjects

___________________________________________________________________________________
SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available

___________________________________________________________________________________
SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2009        DATASTRUC       [3]     ____CORE        4               Available
Enter the Subject Code to Deregister
14cs2054
DearSUHAAS
You are successfully "DEREGISTERED" for C# PROGRM
________________________________________________________________________________
|__________________________|SUBJECT REGISTRATION|_______________________________|
1.Register Subject
2.Deregister Subject
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**]