DIGITAL ASSIGNMENT – 1

Name – Rajvardhan Patil

Reg No – 21BCI0056

Date – 05-09-2022

Q1]

Code:
```
#include <stdio.h>

int main()
{
    //Initialize array
    int arr[] = {1, 2, 3, 4, 5};
    //Calculate length of array arr
    int length = sizeof(arr)/sizeof(arr[0]);
    //n determine the number of times an array should be rotated
    int n = 3;

    //Displays original array
    printf("Original array: \n");
    for (int i = 0; i < length; i++) {
        printf("%d ", arr[i]);
    }

    //Rotate the given array by n times toward right
    for(int i = 0; i < n; i++){
        int j, last;
        //Stores the last element of the array
        last = arr[length-1];

        for(j = length-1; j > 0; j--){
            //Shift element of array by one
            arr[j] = arr[j-1];
        }
        //Last element of array will be added to the start of array.
        arr[0] = last;
    }

    printf("\n");

    //Displays resulting array after rotation
    printf("Array after right rotation: \n");
    for(int i = 0; i< length; i++){
        printf("%d ", arr[i]);

    }
    printf("Reg No - 21BCI0056");
    return 0;
}
```

Output:

```
Original array:
1 2 3 4 5
Array after right rotation:
3 4 5 1 2
Reg No - 21BCI0056
```

Q2]

Code: (If present)

```
// C++ program to check if a string is
// substring of other.
#include <bits/stdc++.h>
using namespace std;

// Returns true if s1 is substring of s2
int isSubstring(string s1, string s2)
{
    int M = s1.length();
    int N = s2.length();

    /* A loop to slide pat[] one by one */
    for (int i = 0; i <= N - M; i++) {
        int j;

        /* For current index i, check for
pattern match */
        for (j = 0; j < M; j++)
            if (s2[i + j] != s1[j])
                break;

        if (j == M)
            return i;
    }

    return -1;
}
```

```
/* Driver code */
int main()
{
    string s1 = "and";
    string s2 = "data struct and algo";
    int res = isSubstring(s1, s2);
    if (res == -1)
        cout << "Not present";
    else
        cout << "Present at index  " << res << "\n"<<s1;
    return 0;
}
```

Output:

1.

```
Present at index  12
and
Reg No - 21BCI0056
```

Code: (Not present)

```
// C++ program to check if a string is
// substring of other.
#include <bits/stdc++.h>
using namespace std;

// Returns true if s1 is substring of s2
int isSubstring(string s1, string s2)
{
    int M = s1.length();
    int N = s2.length();

    /* A loop to slide pat[] one by one */
    for (int i = 0; i <= N - M; i++) {
        int j;

        /* For current index i, check for
pattern match */
        for (j = 0; j < M; j++)
            if (s2[i + j] != s1[j])
                break;

        if (j == M)
```

```
            return i;
    }

    return -1;
}

/* Driver code */
int main()
{
    string s1 = "for";
    string s2 = "data struct and algo";
    int res = isSubstring(s1, s2);
    if (res == -1)
        cout << "Not present";
    else
        cout << "Present at index  " << res << "\n"<<s1;
    return 0;
}
```

Output:

```
Not present
Reg No - 21BCI0056
```

Q3]

Code:

```
#include <stdio.h>
#include <string.h>

#define max 100
int top,stack[max];

void push(char x){

      // Push(Inserting Element in stack) operation
      if(top == max-1){
          printf("stack overflow");
      }  else {
          stack[++top]=x;
      }

}

void pop(){
    // Pop (Removing element from stack)
      printf("%c",stack[top--]);
}


 int main()
{
   char str[]="Data Structure";
   int len = strlen(str);
   int i;
    printf("Reg No - 21BCI0056\n");

   for(i=0;i<len;i++)
        push(str[i]);

   for(i=0;i<len;i++)
      pop();
}
```

Output:

```
Reg No - 21BCI0056
erutcurtS ataD
```

Q4]

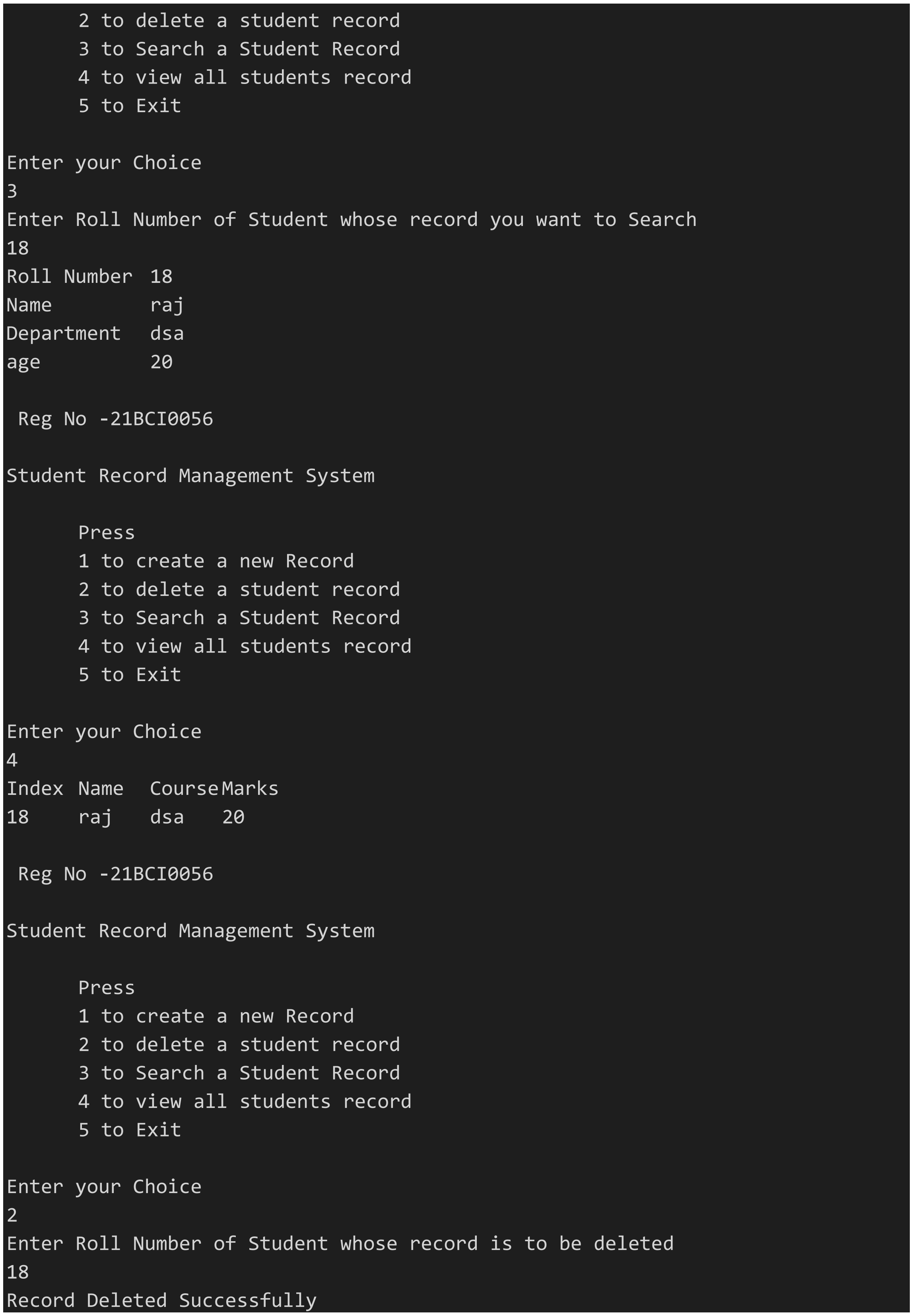
Code:

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
  int roll;
  string Name;
  string Dept;
  int age;
  Node *next;
};

Node *head = new Node();

bool check(int x)
{
  if (head == NULL)
```

```
    return false;

  Node *t = new Node;
  t = head;

  while (t != NULL)
  {
    if (t->roll == x)
      return true;
    t = t->next;
  }

  return false;
}

void Insert_Record(int roll, string Name,
                   string Dept, int age)
{
  if (check(roll))
  {
    cout << "Student with this "
         << "record Already Exists\n";
    return;
  }

  Node *t = new Node();
  t->roll = roll;
  t->Name = Name;
  t->Dept = Dept;
  t->age = age;
  t->next = NULL;

  if (head == NULL || (head->roll >= t->roll))
  {
    t->next = head;
    head = t;
  }

  else
  {
    Node *c = head;
    while (c->next != NULL && c->next->roll < t->roll)
    {
      c = c->next;
    }
    t->next = c->next;
    c->next = t;
  }
  cout << "Record Inserted "
       << "Successfully\n";
}

void Search_Record(int roll)
{
  if (!head)
  {
    cout << "No such Record "
         << "Available\n";
    return;
  }

  else
  {
    Node *p = head;
    while (p)
    {
      if (p->roll == roll)
      {
        cout << "Roll Number\t"
             << p->roll << endl;
        cout << "Name\t\t"
             << p->Name << endl;
        cout << "Department\t"
             << p->Dept << endl;
        cout << "age\t\t"
             << p->age << endl;
        return;
      }
      p = p->next;
    }

    if (p == NULL)
      cout << "No such Record "
           << "Available\n";
  }
}

int Delete_Record(int roll)
{
  Node *t = head;
  Node *p = NULL;

  // Deletion at Begin
  if (t != NULL && t->roll == roll)
  {
    head = t->next;
    delete t;

    cout << "Record Deleted "
         << "Successfully\n";
    return 0;
  }

  // Deletion Other than Begin
  while (t != NULL && t->roll != roll)
  {
    p = t;
    t = t->next;
  }
  if (t == NULL)
  {
    cout << "Record does not Exist\n";
    return -1;
    p->next = t->next;

    delete t;
    cout << "Record Deleted "
         << "Successfully\n";

    return 0;
  }
};

void Show_Record()
{
  Node *p = head;
  if (p == NULL)
  {
    cout << "No Record "
         << "Available\n";
  }
  else
  {
    cout << "Index\tName\tCourse"
         << "\tMarks\n";

    while (p != NULL)
    {
      cout << p->roll << " \t"
           << p->Name << "\t"
           << p->Dept << "\t"
           << p->age << endl;
      p = p->next;
    }
  }
}

int main()
{
  head = NULL;
  string Name, Course;
  int Roll, age;

  while (true)
  {
    cout << "\n Reg No -21BCI0056\n\t\nStudent Record "
            "Management System\n\n\tPress\n\t1 to "
            "create a new Record\n\t2 to delete a "
            "student record\n\t3 to Search a Student "
            "Record\n\t4 to view all students "
            "record\n\t5 to Exit\n";
    cout << "\nEnter your Choice\n";
    int Choice;

    cin >> Choice;
    if (Choice == 1)
    {
      cout << "Enter Name of Student\n";
      cin >> Name;
      cout << "Enter Roll Number of Student\n";
      cin >> Roll;
      cout << "Enter Course of Student \n";
      cin >> Course;
      cout << "Enter Total age of Student\n";
      cin >> age;
      Insert_Record(Roll, Name, Course, age);
    }
    else if (Choice == 2)
    {
      cout << "Enter Roll Number of Student whose "
              "record is to be deleted\n";
      cin >> Roll;
      Delete_Record(Roll);
    }
    else if (Choice == 3)
    {
      cout << "Enter Roll Number of Student whose "
              "record you want to Search\n";
      cin >> Roll;
      Search_Record(Roll);
    }
    else if (Choice == 4)
    {
      Show_Record();
    }
    else if (Choice == 5)
    {
      exit(0);
    }
    else
    {
      cout << "Invalid Choice "
           << "Try Again\n";
    }
  }
  return 0;
}
```

Output:

```
Reg No -21BCI0056

Student Record Management System

      Press
      1 to create a new Record
      2 to delete a student record
      3 to Search a Student Record
      4 to view all students record
      5 to Exit

Enter your Choice
1
Enter Name of Student
raj
Enter Roll Number of Student
18
Enter Course of Student
dsa
Enter Total age of Student
20
Record Inserted Successfully

 Reg No -21BCI0056

Student Record Management System

      Press
      1 to create a new Record
```



```
      2 to delete a student record
      3 to Search a Student Record
      4 to view all students record
      5 to Exit

Enter your Choice
3
Enter Roll Number of Student whose record you want to Search
18
Roll Number  18
Name         raj
Department   dsa
age          20

 Reg No -21BCI0056

Student Record Management System

      Press
      1 to create a new Record
      2 to delete a student record
      3 to Search a Student Record
      4 to view all students record
      5 to Exit

Enter your Choice
4
Index Name   CourseMarks
18    raj    dsa   20

 Reg No -21BCI0056

Student Record Management System

      Press
      1 to create a new Record
      2 to delete a student record
      3 to Search a Student Record
      4 to view all students record
      5 to Exit

Enter your Choice
2
Enter Roll Number of Student whose record is to be deleted
18
Record Deleted Successfully
```

Q5]

Code:

```
#include<stdio.h>
#include<ctype.h>

char stack[100];
int top = -1;

void push(char x)
{
   stack[++top] = x;
}

char pop()
{
   if(top == -1)
      return -1;
   else
      return stack[top--];
}

int priority(char x)
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
   return 0;
}

int main()
{
   char exp[100];
   char *e, x;
   printf("Reg No - 21BCI0056\n");
   printf("Enter the expression : ");
   scanf("%s",exp);
   printf("\n");
   e = exp;

   while(*e != '\0')
   {
      if(isalnum(*e))
         printf("%c ",*e);
      else if(*e == '(')
         push(*e);
      else if(*e == ')')
      {
         while((x = pop()) != '(')
            printf("%c ", x);
      }
      else
      {
         while(priority(stack[top]) >= priority(*e))
            printf("%c ",pop());
         push(*e);
      }
      e++;
   }

   while(top != -1)
   {
      printf("%c ",pop());
   }return 0;
}
```

Output:

```
Reg No - 21BCI0056
Enter the expression : (a+b)/(a*b)+k
a b + a b * / k +
```

Q6]

Code:

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int precedence(char op){
    if(op == '+'||op == '-')
    return 1;
    if(op == '*'||op == '/')
    return 2;
    return 0;
}

int applyOp(int a, int b, char op){
    switch(op){
        case '+': return a + b;
        case '-': return a - b;
        case '*': return a * b;
        case '/': return a / b;
    }
}

int evaluate(string tokens){
    int i;

    stack <int> values;

    stack <char> ops;

    for(i = 0; i < tokens.length(); i++){

        if(tokens[i] == ' ')
            continue;

        else if(tokens[i] == '('){
            ops.push(tokens[i]);
        }

        else if(isdigit(tokens[i])){
            int val = 0;

            while(i < tokens.length() &&
                        isdigit(tokens[i]))
            {
                val = (val*10) + (tokens[i]-'0');
                i++;
            }

            values.push(val);
            i--;
        }


        else if(tokens[i] == ')')
        {
            while(!ops.empty() && ops.top() != '(')
            {
                int val2 = values.top();
                values.pop();

                int val1 = values.top();
                values.pop();

                char op = ops.top();
                ops.pop();

                values.push(applyOp(val1, val2, op));
            }


            if(!ops.empty())
            ops.pop();
        }


        else
        {

            while(!ops.empty() && precedence(ops.top())
                                >= precedence(tokens[i])){
                int val2 = values.top();
                values.pop();

                int val1 = values.top();
                values.pop();

                char op = ops.top();
                ops.pop();

                values.push(applyOp(val1, val2, op));
            }


            ops.push(tokens[i]);
        }
```

```
    }


    while(!ops.empty()){
        int val2 = values.top();
        values.pop();

        int val1 = values.top();
        values.pop();

        char op = ops.top();
        ops.pop();

        values.push(applyOp(val1, val2, op));
    }


    return values.top();
}

int main() {
    printf("Reg No - 21BCI0056\n");
    cout << evaluate("10 + 2 * 6") << "\n";
    cout << evaluate("100 * 2 + 12") << "\n";
    cout << evaluate("100 * ( 2 + 12 )") << "\n";
    cout << evaluate("100 * ( 2 + 12 ) / 14");
    return 0;
}
```

Output:

```
Reg No - 21BCI0056
22
212
1400
100
```

Q7]

Code:

```
#include <bits/stdc++.h>

using namespace std;
```

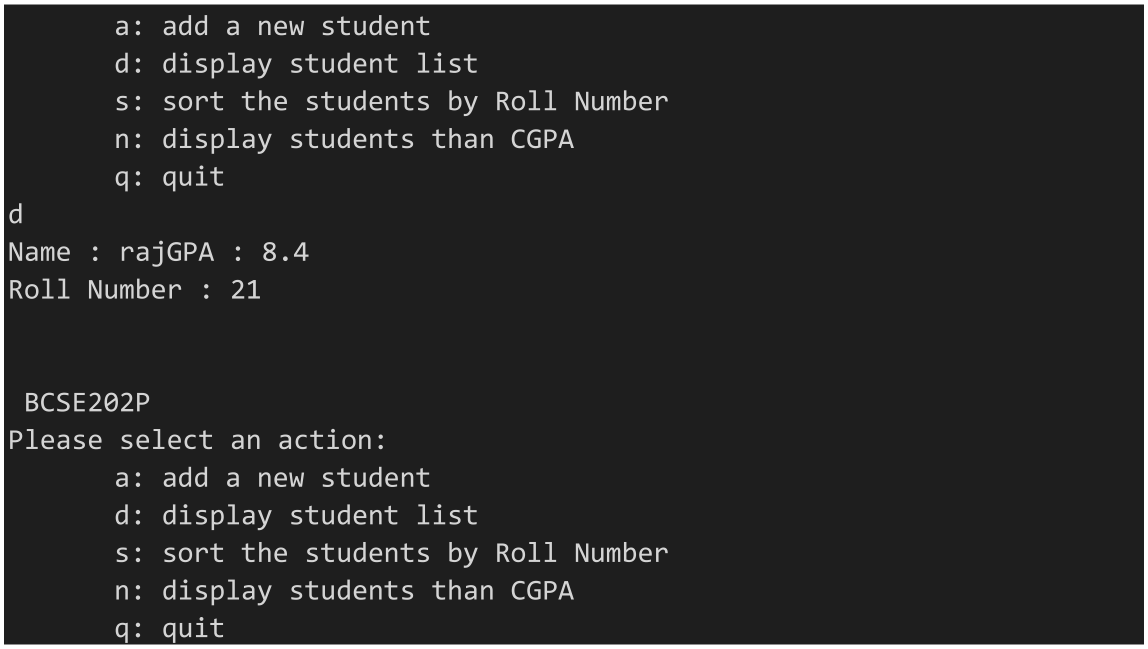
```
class Student
{
private:
  string name;
  float gpa;
  int rollNumber;

public:
  Student();
  void setName(string name_input);
  void setGpa(float gpa_input);
  void setRollNumber(int rollNumber_input);
  void displayStudent();
  string getName();
  float getGpa();
  int getRollNumber();
};
Student::Student()
{
  name = "abc";
  gpa = 1.0;
  rollNumber = 0;
}
void Student::setName(string name_input)
{
  name = name_input;
}
void Student::setGpa(float gpa_input)
{
  gpa = gpa_input;
}
void Student::setRollNumber(int rollNumber_input)
{
  rollNumber = rollNumber_input;
}
void Student::displayStudent()
{
  cout << "Name : " << name;
  cout << "GPA : " << gpa << endl;
  cout << "Roll Number : " << rollNumber << endl;
}
string Student::getName()
{
  return name;
}
float Student::getGpa()
```

```
{
  return gpa;
}
int Student::getRollNumber()
{
  return rollNumber;
}
#define MAX_STUDENTS 5
void executeAction(char);
int addStudent(string name_input, float gpa_input, int
rollNumber_input);
void displayStudents();
void sort();
void studentsAfterGivenYear();
Student s[MAX_STUDENTS];
int currentCount = 0;
int main()
{
  char choice = 'i';
  do
  {
    cout << "\n BCSE202P \n";
    cout << "Please select an action:\n";
    cout << "\t a: add a new student\n";
    cout << "\t d: display student list\n";
    cout << "\t s: sort the students by Roll Number\n";
    cout << "\t n: display students than CGPA\n";
    cout << "\t q: quit\n";
    cin >> choice;
    cin.ignore();
    executeAction(choice);
  } while (choice != 'q');
  return 0;
}
void executeAction(char c)
{
  string name_input;
  float gpa_input;
  int rollNumber_input, result = 0;
  switch (c)
  {
  case 'a': // add student
    // input student details from use
    cout << "Please enter student name: ";
    getline(cin, name_input);
    cout << "Please enter GPA: ";
    cin >> gpa_input;
    cin.ignore();
    cout << "Please enter roll number: ";
    cin >> rollNumber_input;
    cin.ignore();
    // add the student to the list
    result = addStudent(name_input, gpa_input, rollNumber_input);
    if (result == 0)
      cout << "\nThat student is already in the list or list is full!
\n\n";
    else
      cout << "\nStudent successfully added to the list! \n\n";
    break;
  case 'd': // display the list
    displayStudents();
    break;
  case 's': // sort the list
    sort();
    break;
  case 'n': // display after given year
    studentsAfterGivenYear();
    break;
  case 'q':
    break;
  default:
    cout << c << " is invalid input!\n";
  }
}
int addStudent(string name_input, float gpa_input, int
rollNumber_input)
{
  if (currentCount < MAX_STUDENTS)
  {
    for (int i = 0; i < currentCount; i++)
      if ((s[i].getName() == name_input) && (s[i].getGpa() ==
gpa_input) && (s[i].getRollNumber() == rollNumber_input))
        return 0;
    Student temp;
    temp.setName(name_input);
    temp.setGpa(gpa_input);
    temp.setRollNumber(rollNumber_input);
    s[currentCount] = temp;
    currentCount++;
    return 1;
  }
  return 0;
```

```
}
void displayStudents()
{
  for (int i = 0; i < currentCount; i++)
  {
    s[i].displayStudent();
    cout << endl;
  }
}
void sort()
{
  Student temp;
  int max;
  for (int i = 0; i < currentCount - 1; i++)
  {
    max = i;
    for (int j = i + 1; j < currentCount; j++)
    {
      if (s[j].getRollNumber() > s[max].getRollNumber())
        max = j;
    }
    if (max != i)
    {
      temp = s[i];
      s[i] = s[max];
      s[max] = temp;
    }
  }
  cout << endl
       << "Student list sorted! Use d option to see the sorted result."
<< endl;
}
void studentsAfterGivenYear()
{
  int cap;
  Student *newStudent = new Student;
  cout << "Enter the cap bound of cgpa : ";
  cin >> cap;
  for (int i = 0; i < currentCount; i++)
  {
    if (s[i].getGpa() >= cap)
    {
      newStudent->setGpa(s[i].getGpa());
      newStudent->setName(s[i].getName());
      newStudent->setRollNumber(s[i].getRollNumber());
      newStudent->displayStudent();
      cout << endl;
    }
  }
}
```

Output:

```
Reg No – 21BCI0056
BCSE202P
Please select an action:
       a: add a new student
       d: display student list
       s: sort the students by Roll Number
       n: display students than CGPA
       q: quit
a
Please enter student name: raj
Please enter GPA: 8.4
Please enter roll number: 21
Student successfully added to the list!


 BCSE202P
Please select an action:
       a: add a new student
       d: display student list
       s: sort the students by Roll Number
       n: display students than CGPA
       q: quit
n
Enter the cap bound of cgpa : 8.6
Name : rajGPA : 8.4
Roll Number : 21


 BCSE202P
Please select an action:
       a: add a new student
       d: display student list
       s: sort the students by Roll Number
       n: display students than CGPA
       q: quit
. is invalid input!

 BCSE202P
Please select an action:
```



```
       a: add a new student
       d: display student list
       s: sort the students by Roll Number
       n: display students than CGPA
       q: quit
d
Name : rajGPA : 8.4
Roll Number : 21


 BCSE202P
Please select an action:
       a: add a new student
       d: display student list
       s: sort the students by Roll Number
       n: display students than CGPA
       q: quit
```