دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

5/4/2020

Report Homework 5

Advanced Programming

Mohammad Javad Amin (9523008)
AMIRKABIR UNIVERSITY OF TECHNOLOGY

به نام خدا

هدف پروژه ؛ تخمین نمره پایانی دانشجویان به وسیله متغییر تصادفی با توزیع احتمال (pdf) است که این توزیع احتمال یکی از دو توزیع نرمال یا نمایی است که پارامتر های آن توسط عملکرد دانشجو در طول ترم مشخص می‌شود.

ابتدا باید یک صف را پیاده‌سازی کنیم که از آن برای نگهداری دانشجوها استفاده می‌شود.

class APQueue : الگوریتم این صف طبق قانون FIFO است یعنی چیزی اول بیاید اول هم می‌رود. توجه شود که این صف باید بتواند هر چیزی را ذخیره کند پس آن را به صورت templateتعریف می‌کنیم.

این class شامل member variable و همچنین member function هایی است که در ادامه به تعریف و بررسی تک تک آن ها می پردازیم.

member variable های class APDS به صورت private :

۱)Item : این خود یک class است که در همین جا(در قسمت private) باید آن را تعریف کنیم . class Item خود دارای member variable و همچنین member function هایی است که آن ها تک تک بررسی بایدکرد .

توجه بایدکرد که برای اینکه بتوان به member variable و همچنین member function های class Item در داخل class APQueue دسترسی پیدا کنیم باید آن‌ها را به صورت public تعریف کرد.

member variable های class Item :

- pvalue : یک متغییر از جنس pointer بوده که در آن آدرس objectذخیره می‌شود.
- pnext : یک متغییر از جنس pointer بوده که در آن آدرس object Item بعدی ذخیره می‌شود.

member function های class Item :

- constructor که داری یک ورودی از جنس objectورودی است که به صورت reference به آن داده می‌شود.

پیاده‌سازی این تابع به صورت زیر است.

```cpp
template<typename T>
APQueue<T>::Item::Item(T& S)
{
    pvalue = &S;            //save object
    pnext = nullptr;        //initialize pnext
}
```

۲)phead : یک متغییر از جنس pointer بوده که در آن آدرس object Item ابتدایی صف (head) در آن ذخیره می‌شود.

۳)ptail : یک متغییر از جنس pointer بوده که در آن آدرس object Item انتهایی صف(tail) در آن ذخیره می‌شود.

۴)size : تعداد دانشجویان داخل data structure .

در زیرکل class APQueue را مشاهده می‌کنیم.

```cpp
template<typename T>
class APQueue
{
private:
    class Item                //class Item
    {
    public:
        T* pvalue;           //current object
        Item* pnext;         //adress next item
        Item(T&);            //constructor class Item
    };

    Item* phead;          //adress of the head in queue
    Item* ptail;          //adress of the tail in queue
    size_t size;          //number of students in data struture

public:
    APQueue(std::vector<T>&);       //constructor
    APQueue(std::vector<T>&&);      //constructor for rvalue
    ~APQueue();                     //destructor

    void enqueue(T&);               //gets a object and push it to queue
    APQueue<T>* enqueue(T&&);       //gets a object rvalue and push it to queue
    T dequeue();                    //removes the first thing in queue
    void show();                    //show detail

    T operator[](size_t i);         //operator []
};
```

member function های class APQueue :

۱)constructor : ورودی آن یک وکتور که به صورت reference به آن داده شده ؛ از چیز هایی است که میخواهم وارد صف کنیم عمل وارد کرد dataها را در صف با تابع enqueue انجام می‌دهیم. این تابع به صورت زیر پیاده‌سازی می‌شود.

```
template<typename T>
APQueue<T>::APQueue(std::vector<T> &input_vector)   //constructor
{
    size = {};          //initialize size
    ptail = nullptr;    //initialize ptail
    phead = nullptr;    //initialize phead

    for (size_t i = 0; i < input_vector.size(); i++)  //push object to queue
        enqueue(input_vector[i]);
}
```

۲)constructor : مانند constructor قبلی است با این تفاوت که ورودی آن rvalue است و به صورت زیر پیاده‌سازی می‌شود.

```
template<typename T>
APQueue<T>::APQueue(std::vector<T> &&input_vector_in) //constructor for rvalue
{
    static std::vector<T> input_vector{input_vector_in}; //to save input object

    size = {};          //initialize size
    ptail = nullptr;    //initialize ptail
    phead = nullptr;    //initialize phead
    for (size_t i = 0; i < input_vector.size(); i++) //push object to queue
        enqueue(input_vector[i]);
}
```

۳)destructor : تمام عضوهای صف را از بین می‌برد و به صورت زیر پیاده سازی می‌شود.

```
template<typename T>
APQueue<T>::~APQueue() //destructor
{                      //remove all item in queue
    if(size)
    {
        Item* temp{ phead };
        Item* temp_next{ temp->pnext };
        for (size_t i = 0; i < size-1; i++)
        {
            delete temp;
            temp = temp_next;
            temp_next = temp->pnext;
        }
        delete ptail;
    }
}
```

۴)enqueue : این تابع یک داده جدید می‌گیرد و آن را به آخر صف اضافه می‌کند.و هر دفعه باید آدرس ها داده های انتهای صف به روزرسانی شود. این تابع به دو صورت تعریف می شود که فرق آن ها بر اساس ورودی آن است که یکی به صورت referencr بوده و دیگری ورودی آن rvalue است و باید طوری پیاده سازی شود که پشت سرهم بتواند این تابع را call کرد. پیاده سازی این دو تابع به صورت زیر است.

```
template<typename T>
void APQueue<T>::enqueue(T& in)    //get object in queue
{
    Item* new_input{ new Item{in} }; //create new item with input

    if (size > 0)
        ptail->pnext = new_input;
    ptail = new_input;
    if (size == 0)   //for save phead
        phead = new_input;
    size++;
}
```

```
template<typename T>
APQueue<T>* APQueue<T>::enqueue(T&& in) //get object in queue for rvalue
{
    Item* new_input{ new Item{in} };    //create new item with input

    if (size > 0)
        ptail->pnext = new_input;
    ptail = new_input;
    if (size == 0)              //for save phead
        phead = new_input;
    size++;

    return this;
}
```

۵)dequeue : این تابع اولین عضو صف (phead) را از صف خارج کرده و آدرس ها را طبق شرایط جدید به روز رسانی می‌کند و در آخر عضو خارج شده را چاپ می‌کند. . این تابع به صورت زیر پیاده‌سازی می‌شود.

```
template<typename T>
T APQueue<T>::dequeue() //removes the first thing in queue
{
    if (size == 0)  //If the queue is empty return 0
    {
        ptail = phead = nullptr;
        std::cout<<"queue is empty"<<std::endl;
        return 0;
    }
    //removes last data and update adress
    Item* temp{ phead->pnext };
    T item{ *(phead->pvalue) };

    delete phead;   //delete phead
    phead = temp;
    size--;    //decrease size queue

    return item;
}
```

۶)show : این تابع تعداد عضو های صف و تک تک اعضا را نمایش می دهد و به صورت زیر پیاده‌سازی می‌شود.

```
template<typename T>
void APQueue<T>::show()       //show detail
{
    Item* temp{ phead };

    std::cout << "APQueue of size: " << size << std::endl;
    for (size_t i = 0; i < size; i++)
    {
        std::cout << std::setw(5) << "NO : " << i + 1 << std::setw(8) << *(temp->pvalue) << std::endl;
        temp = temp->pnext;
    }
}
```

۷) operator[] : عضو i ام از صف را برمی‌گرداند. این تابع به صورت زیر پیاده‌سازی می‌شود.

```
template<typename T>    //operator[]
T APQueue<T>::operator[](size_t j)
{
    Item* temp{ phead };

    for (size_t i = 0; i < size; i++)
    {
        if (!(i - j))
            return *(temp->pvalue);
        temp = temp->pnext;
    }
    std::cout<<"object not fined!"<<std::endl;  //if i-th is out of the range
    return 0;
}
```

در مرحله بعد این کلاس را برای ورودی با جنس unique_ptr ، به صورت جدا تعریف می‌کنیم.(برای قسمت آخر سوال)

پیاده سازی این class دقیقا به صورت قبلی است با این تفاوت که ورودی آن مشخص بوده .

```
template <>                     //specializations
class APQueue<std::unique_ptr<Student>>
{
private:
    class Item                  //class Item
    {
    public:
        Student* pvalue;        //current student
        Item* pnext;            //adress next item
        inline Item(Student*&); //constructor class Item
    };

    Item* phead;                //adress of the head in queue
    Item* ptail;                //adress of the tail in queue
    size_t size;                //number of students in data struture

public:
    inline APQueue(std::vector<std::unique_ptr<Student>>&&); //constructor for rvalue
    inline~APQueue();                                        //destructor
    inline void enqueue(Student*&);                          //gets a object and push it to queue
    inline void show();                                      //show detail
};
```

پیاده سازی constructor وdestructor و enqueue دقیقا مثل قبل است فقط باید توجه کرد که قبل از اسم توابع inline را برای جلوگیری از خطا نوشت .

```
//specializations

inline APQueue<std::unique_ptr<Student>>::Item::Item(Student*& S)
{
    pvalue = S;        //save object
    pnext = nullptr;  //initialize pnext
}

inline APQueue<std::unique_ptr<Student>>::APQueue(std::vector<std::unique_ptr<Student>>&& input_vector_in)   //constructor for rvalue
{
    static std::vector<Student*> input_vector(input_vector_in.size(),nullptr);     //to save input object

    size = {};         //initialize size
    ptail = nullptr;   //initialize ptail
    phead = nullptr;   //initialize phead
    for (size_t i = 0; i < input_vector.size(); i++)
    {
        input_vector[i] = input_vector_in[i].release();  //save object
        enqueue(input_vector[i]);    //get in queue
    }
}
```

```
inline APQueue<std::unique_ptr<Student>>::~APQueue()   //destructor
{                                                      //remove all item in queue
    if (size)
    {
        Item* temp{ phead };
        Item* temp_next{ temp->pnext };
        for (size_t i = 0; i < size - 1; i++)
        {
            delete temp;
            temp = temp_next;
            temp_next = temp->pnext;
        }
        delete ptail;
    }
}

inline void APQueue<std::unique_ptr<Student>>::enqueue(Student*& in)  //get object in queue
{
    Item* new_input{ new Item{in} };  //create new item with input

    if (size > 0)
        ptail->pnext = new_input;
    ptail = new_input;
    if (size == 0)   //save phead
        phead = new_input;
    size++;
}
```

تابع show در این حالت صف را به نحو دیگر نمایش میدهد و به صورت زیر پیاده‌سازی می‌شود.

```
inline void APQueue<std::unique_ptr<Student>>::show()  //show detail
{
    Item* temp{ phead };

    std::cout << "APQueue of size: " << size << std::endl;
    for (size_t i{}; i < size; i++)
    {
        std::cout << std::setw(5) << "NO : " << i + 1 << std::setw(6) << ", Name: " << temp->pvalue->getName() << std::endl;
        std::cout << "\t\t  " << "Pr{A Grade} =" << std::setprecision(5) << std::fixed << temp->pvalue->between(17, 20);
        std::cout << "\t\t  " << "Pr{B Grade} =" << temp->pvalue->between(14, 17);
        std::cout << "\t\t  " << "Pr{C Grade} =" << temp->pvalue->between(10, 14);
        std::cout << "\t\t  " << "Pr{D Grade} =" << temp->pvalue->between(0, 10) << std::endl;
        temp = temp->pnext;
    }
}
```

در قسمت بعدی class Student را تعریف می‌کنیم و به صورت base بوده و توابع آن به صورت pure virtual اند و باید این توابع در class هایی که از class Student ارث می‌برند به صورت اختصاصی تعریف شوند.

این class شامل member variable و همچنین member function هایی است که در ادامه به تعریف و بررسی تک تک آن ها می پردازیم.

member variable ها به صورت private :

۱)name :نام دانشجو

member function های class Student که به صورت pure virtualاند :

۱)pdf : این تابع یک عدد از ورودی گرفته و احتمال این که دانشجو این نمره را کسب کند بازمیگرداند.

۲)integrator : این تابع سه ورودی a,b,step_size را می‌گیرد سپس انتگرال تابع pdf از نقطه a تاb را به روش ذوزنقه ایی حساب می‌کند.

۳)between : این تابع به ترتیب ۲ ورودی a و b می گیرد و احتمال اینکه نمره دانشجو بین a تا b شود را توسط انتگرال گیری عددی به روش ذوزنقه ای از تابع pdf در بازه a تا b محاسبه می کند.

پیاده سازی این class به صورت زیر است.

```
class Student
{
protected:
    std::string name{"MJ amin"};   //name student
public:
    virtual double pdf(double x)=0;                    //pdf function
    virtual double integrator(double a, double b, double step_size = 0.001) = 0;  //integrator
    virtual double between(double a, double b) = 0;   //probablity between a , b
    virtual std::string getName() = 0;                 //get name
};
```

قسمت بعدی تعریف class NoramlStudent است که از class Student ارث می‌برد.

این class شامل member variable و همچنین member function هایی است که در ادامه به تعریف و بررسی تک تک آن ها می پردازیم.

member variable ها به صورت private :

۱)mean :میانگین توزیع نرمال

۲)std : انحراف معیار توزیع نرمال

در زیرکل class NoramlStudent را مشاهده می‌کنیم.

```
class NormalStudent : public Student
{
private:
    double mean;    //mean
    double std;     //std
public:
    NormalStudent(std::string name, double mean = 12, double std = 2);  //constructor
    NormalStudent();   //default constructor

    double pdf(double x);                       //pdf function
    double integrator(double a, double b, double step_size = 0.001);  //integrator function
    double between(double a, double b);         //probablity between a , b
    double operator()(double x);                //operator() return pdf(x)
    double operator()(double a, double b);      //operator() return between(a,b)
    std::string getName();                      //get name
};
```

member function های class NoramlStudent :

۱)constructor : شال ورودی های زیر است.

- name :نام دانشجو
- mean :میانگین توزیع نرمال
- std : انحراف معیار توزیع نرمال

```
NormalStudent::NormalStudent(std::string name, double mean, double std) //constructor
{
    this->mean = mean;    //save mean
    this->std = std;      //save std
    this->name = name;    //save name
}
```

۲) default constructor: به صورت زیر پیاده‌سازی می‌شود.

```
NormalStudent::NormalStudent()  //copy constructor
{
    mean = 12;                  //save mean
    std = 2;                    //save std
    name = { "mj amin" };       //save name
}
```

در این class باید توابع موجود در class Student را به صورت اختصاصی تعریف کنیم.

۳)pdf : این تابع یک عدد از ورودی گرفته و احتمال این که دانشجو این نمره را کسب کند بازمیگرداند. این تابع به صورت زیر پیاده سازی می شود.

$$pdf(x) = \frac{1}{\sqrt{2\pi}(\text{std})} e^{-\frac{1}{2(\text{std})^2}(x-\text{mean})^2}$$

```
double NormalStudent::pdf(double x) //pdf function
{
    const double pi{ 3.14159 };

    return (1 / (std::sqrt(2 * pi) * std)) * exp(-(1 / (2 * std * std)) * (x - mean) * (x - mean));
}
```

۴)integrator : این تابع سه ورودی a,b,step_size را می‌گیرد سپس انتگرال تابع pdf از نقطه a تاb را به روش ذوزنقه ایی حساب می‌کند و به صورت زیر پیاده‌سازی می‌شود.

$$\mathrm{A} = \int_a^b f(x)\,dx \approx \Delta x\left(\frac{y_0}{2} + y_1 + \ldots + \frac{y_n}{2}\right)$$

```
double NormalStudent::integrator(double a, double b, double step_size)  //integtator
{
    double sum{};
    double temp{ a };                                   //start from a

    for (size_t step{}; step <= (b - a) / step_size; step ++)
    {
        if (step == 0 || step == (b - a) / step_size)
            sum += pdf(temp) / 2;
        else
            sum += pdf(temp);
        temp += step_size;
    }

    return step_size*sum;                               //retrnu integrator (a,b)
}
```

۵)between : این تابع به ترتیب ۲ ورودی a و b می گیرد و احتمال اینکه نمره دانشجو بین a تا b شود را توسط انتگرال گیری عددی به روش ذوزنقه ای از تابع pdf در بازه a تا b محاسبه می کند. این تابع به صورت زیر پیاده سازی می شود.

```
double NormalStudent::between(double a, double b)       //probablity between a , b
{
    return integrator(a,b);
}
```

۶)operator() : به دو صورت پیاده سازی می شود

- اگر شامل یک ورودی بود باید تابع pdf را call کند
- اگر شامل دو وردی بود باید تابع between را call کند.

این توابع به صورت زیر پیاده می شود.

```
double NormalStudent::operator()(double x)              //operator() retuen pdf(x)
{
    return pdf(x);
}

double NormalStudent::operator()(double a, double b)    //operator()  return between(a,b)
{
    return between(a,b);
}
```

۷)getName : اسم دانشجو را برمی‌گرداند و به صورت زیر پیاده سازی می‌شود.

```
std::string NormalStudent::getName()                    //get name
{
    return name;
}
```

قسمت بعدی تعریف class ExponentialStudent است که از class Student ارث می‌برد

این class شامل member variable و همچنین member function هایی است که در ادامه به تعریف و بررسی تک تک آن ها می پردازیم.

member variable ها به صورت private :

۱)lambda :پارامتر توزیع احتمال نمایی

در زیرکل class ExponentialStudent را مشاهده می‌کنیم.

```
class ExponentialStudent : public Student
{
private:
    double lambda;      //lambda
public:
    ExponentialStudent(std::string name, double lambda = 0.07);  //constructor
    ExponentialStudent();                                        //defual constructor

    double pdf(double x);                                        //pdf function
    double integrator(double a, double b, double step_size = 0.001);  //integrator function
    double between(double a, double b);                          //probablity between a , b
    double operator()(double x);                                 //operator() retuen pdf(x)
    double operator()(double a, double b);                       //operator()  return between(a,b)
    std::string getName();                                       //get name

};
```

member function های class ExponentialStudent :

۱)constructor : شال ورودی های زیر است.

- name :نام دانشجو
- lambda :پارامتر توزیع احتمال نمایی

```
NormalStudent::NormalStudent(std::string name, double mean, double std) //constructor
{
    this->mean = mean;   //save mean
    this->std = std;     //save std
    this->name = name;   //save name
}
```

۲) copy constructor: به صورت زیر پیاده‌سازی می‌شود.

```
NormalStudent::NormalStudent()  //default constructor
{
    mean = 12;                  //save mean
    std = 2;                    //save std
    name = { "mj amin" };       //save name
}
```

در این class باید توابع موجود در class Student را به صورت اختصاصی تعریف کنیم.

۳)pdf : این تابع یک عدد از ورودی گرفته و احتمال این‌که دانشجو این نمره را کسب کند بازمیگرداند. این تابع به صورت زیر پیاده سازی می شود.

$$pdf(x) = (lambda)e^{-(lambda)x}$$

```
double NormalStudent::pdf(double x) //pdf function
{
    const double pi{ 3.14159 };

    return (1 / (std::sqrt(2 * pi) * std)) * exp(-(1 / (2 * std * std)) * (x - mean) * (x - mean));
}
```

۴)integrator : این تابع سه ورودی a,b,step_size را می‌گیرد سپس انتگرال تابع pdf از نقطه a تاb را به روش ذوزنقه ایی حساب می‌کند و به صورت صفحه بعد پیاده‌سازی می‌شود.

$$\mathrm{A} = \int_a^b f(x)\,dx \approx \Delta x(\frac{y_0}{2} + y_1 + \ldots + \frac{y_n}{2})$$

```cpp
double NormalStudent::integrator(double a, double b, double step_size)  //integtator
{
    double sum{};
    double temp{ a };                                   //start from a

    for (size_t step{}; step <= (b - a) / step_size; step ++)
    {
        if (step == 0 || step == (b - a) / step_size)
            sum += pdf(temp) / 2;
        else
            sum += pdf(temp);
        temp += step_size;
    }

    return step_size*sum;                               //retrnu integrator (a,b)
}
```

۵)between : این تابع به ترتیب ۲ ورودی a و b می گیرد و احتمال اینکه نمره دانشجو بین a تا b شود را توسط انتگرال گیری عددی به روش ذوزنقه ای از تابع pdf در بازه a تا b محاسبه می کند. این تابع به صورت زیر پیاده سازی می شود.

```cpp
double NormalStudent::between(double a, double b)       //probablity between a , b
{
    return integrator(a,b);
}
```

۶)operator() : به دو صورت پیاده سازی می شود

- اگر شامل یک ورودی بود باید تابع pdf را call کند
- اگر شامل دو وردی بود باید تابع between را call کند.

این توابع به صورت زیر پیاده می شود.

```cpp
double NormalStudent::operator()(double x)              //operator() retuen pdf(x)
{
    return pdf(x);
}

double NormalStudent::operator()(double a, double b)    //operator()  return between(a,b)
{
    return between(a,b);
}
```

۷)getName : اسم دانشجو را برمی‌گرداند و به صورت زیر پیاده سازی می‌شود.

```cpp
std::string NormalStudent::getName()                    //get name
{
    return name;
}
```

بعد از اتمام تعریف class ها حال به سراغ تابع GetData را تعریف کنیم.

GetData : ورودی این تابع اسم فایلی است که dataها در آن قرار دارند و کار این تابع خواندن فایل و ذخیره کردن اطلاعات هر دانشجو و برگرداندن کل دانشجویان به صورت وکتور است. الگوریتم این تابع را در پروژه های پیش شرح داده ایم. پیاده‌سازی این تابع به صورت زیر است.

```
std::vector<std::unique_ptr <Student>> getData(const char* filename)    //get data from .csv file
{
    std::vector<std::unique_ptr <Student>> output{};  //for store students
    std::ifstream input_file{ filename };             //read .csv file

    while (!input_file.eof())                         //while until end of file
    {
        std::string student{};                        //for store each student(line of data)
        std::string data{};                           //for store each data of student in a row
        std::unique_ptr <Student> input_student{};    //for save student in each row
        size_t j{};

        std::getline(input_file, student);            //store each student(line)

        if (static_cast<int>(student[0] == 0))        //if line is empty break from loop
            break;

        std::string name;       //name student
        double x{}, y{};
        bool normal{ false };   //show it's normal or not

        for (size_t i{}; i < student.size(); i++)   //to get data from a line
        {
            if (student[i] != ',')
                data += student[i];
            else
            {
                if (j == 0)          //set name
                    name = data;
                else if (j == 1)     //set a
                {
                    x = std::stod(data);
                    normal = true;   //it's normal Steudent
                }
                data = "";
                j++;
            }
        }

        y = std::stod(data);    //set b

        if (normal)
        {
            NormalStudent temp{ name,x,y };        //save this normal Student
            output.push_back(std::make_unique<NormalStudent>(temp));
        }
        else
        {
            ExponentialStudent temp{ name,y };    //save this exponentialStudent
            output.push_back(std::make_unique<ExponentialStudent>(temp));
        }
    }

    return output;
}
```

توجه۱ : در ورودی بعضی از تابع ها ؛ دلیل اینکه ورودی را به صورت refernce می‌دهیم این است که باعث افزایش performance و صرفه‌جویی در حافظه می‌شود. و در برخی موارد ورودی را به صورت const هم می‌دهیم که این کار برای جلوگیری از اجازه تغییر در تابع است.

توجه۲: توضیحات مربوط به پیاده سازی الگوریتم ها در داخل کد به صورت کامنت نوشته شده است و در این گزارش کار سعی شده در مورد الگوریتم توابع توضیح داده شود.

نتایج google test به شرح زیر است.

```
[       OK ] APMidtermTest.APQueueTest3 (0 ms)
[ RUN      ] APMidtermTest.APQueueTest4
[       OK ] APMidtermTest.APQueueTest4 (0 ms)
[ RUN      ] APMidtermTest.APQueueTest5
[       OK ] APMidtermTest.APQueueTest5 (0 ms)
[ RUN      ] APMidtermTest.APQueueTest6
APQueue of size: 0
                                        ********minus 5 points if did not print empty queue!
[       OK ] APMidtermTest.APQueueTest6 (0 ms)
[ RUN      ] APMidtermTest.APQueueTest7
queue is empty
queue is empty
APQueue of size: 1
NO : 1       5
[       OK ] APMidtermTest.APQueueTest7 (0 ms)
[ RUN      ] APMidtermTest.getDataTest
[       OK ] APMidtermTest.getDataTest (0 ms)
[ RUN      ] APMidtermTest.APQueueTest8
APQueue of size: 8
NO : 1, Name: Moradi
                Pr{A Grade} =0.10687            Pr{B Grade} =0.38493            Pr{C Grade} =0.44520            Pr{D Grade} =0.05480
NO : 2, Name: Karimi
                Pr{A Grade} =0.02275            Pr{B Grade} =0.81859            Pr{C Grade} =0.15866            Pr{D Grade} =0.00000
NO : 3, Name: Mahikhar
                Pr{A Grade} =0.02493            Pr{B Grade} =0.02569            Pr{C Grade} =0.03548            Pr{D Grade} =0.09516
NO : 4, Name: Majedinia
                Pr{A Grade} =0.05379            Pr{B Grade} =0.05938            Pr{C Grade} =0.08890            Pr{D Grade} =0.28108
NO : 5, Name: Ghahramani
                Pr{A Grade} =0.11791            Pr{B Grade} =0.11791            Pr{C Grade} =0.14012            Pr{D Grade} =0.19740
NO : 6, Name: Keihani
                Pr{A Grade} =0.81859            Pr{B Grade} =0.02275            Pr{C Grade} =0.00000            Pr{D Grade} =0.00000
NO : 7, Name: Hassanzade
                Pr{A Grade} =0.05954            Pr{B Grade} =0.06917            Pr{C Grade} =0.10995            Pr{D Grade} =0.39347
NO : 8, Name: Amini
                Pr{A Grade} =0.53281            Pr{B Grade} =0.28579            Pr{C Grade} =0.02272            Pr{D Grade} =0.00003
                                        ********minus 10 points if did not print properly!
[       OK ] APMidtermTest.APQueueTest8 (34 ms)
[ RUN      ] APMidtermTest.APQueueTest9
APQueue of size: 8
NO : 1, Name: Moradi
                Pr{A Grade} =0.10687            Pr{B Grade} =0.38493            Pr{C Grade} =0.44520            Pr{D Grade} =0.05480
NO : 2, Name: Karimi
                Pr{A Grade} =0.02275            Pr{B Grade} =0.81859            Pr{C Grade} =0.15866            Pr{D Grade} =0.00000
NO : 3, Name: Mahikhar
                Pr{A Grade} =0.02493            Pr{B Grade} =0.02569            Pr{C Grade} =0.03548            Pr{D Grade} =0.09516
NO : 4, Name: Majedinia
                Pr{A Grade} =0.05379            Pr{B Grade} =0.05938            Pr{C Grade} =0.08890            Pr{D Grade} =0.28108
NO : 5, Name: Ghahramani
                Pr{A Grade} =0.11791            Pr{B Grade} =0.11791            Pr{C Grade} =0.14012            Pr{D Grade} =0.19740
NO : 6, Name: Keihani
                Pr{A Grade} =0.81859            Pr{B Grade} =0.02275            Pr{C Grade} =0.00000            Pr{D Grade} =0.00000
NO : 7, Name: Hassanzade
                Pr{A Grade} =0.05954            Pr{B Grade} =0.06917            Pr{C Grade} =0.10995            Pr{D Grade} =0.39347
NO : 8, Name: Amini
                Pr{A Grade} =0.53281            Pr{B Grade} =0.28579            Pr{C Grade} =0.02272            Pr{D Grade} =0.00003
                        ********minus 10 points if did not print properly with 8 students!
[       OK ] APMidtermTest.APQueueTest9 (40 ms)
[ RUN      ] APMidtermTest.StudentTest
[       OK ] APMidtermTest.StudentTest (1 ms)
[----------] 13 tests from APMidtermTest (98 ms total)

[----------] Global test environment tear-down
[==========] 13 tests from 1 test suite ran. (100 ms total)
[  PASSED  ] 13 tests.
```

همانطورکه قابل مشاهده است نتایج google test موفقیت آمیز بوده.

تمام