دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

4/29/2020

Report Homework 4

Advanced Programming

Mohammad Javad Amin (9523008)
AMIRKABIR UNIVERSITY OF TECHNOLOGY

به نام خدا

هدف پروژه ؛ پیاده‌سازی یک data structure مختص درس AP است . ابتدا ما نمرات تمرین ها و نمرات میان ترم و ... هر دانشجو را وارد میکنیم سپس data structure این دانشجویان را با توجه به ظرفیت خود (capacity) در خود جای داده و آن ها به ترتیب معیار value که برای آن تعریف می‌شود آن ها را مرتب و نمایش می‌دهد.

برای تحقق این هدف نیاز به ۲، class می‌باشد.

اولین class Student است ؛ که شامل نمرات مختلف یک دانشجو است.

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)id :شماره دانشجویی.

۲)homework : نمره تمرین.

۳)midterm_project : نمره‌ی پروژه‌ي میان ترم.

۴)midterm_exam : نمره‌ي میان‌ترم.

۵)party : میزان پارتی دانشجو با TA.

در زیرکل class Student را مشاهده می‌کنیم.

```
class Student
{
private:                    //member variables
    long id{};              //student's id
    double homework{};      //student's homeworks grade
    double midterm_project{};//student's midterm project grade
    double midterm_exam{};  //student's midterm exam grade
    double party{};         //student's amount of friendship with the TAs
public:                     //member functions
    Student(long id, double hw, double mid_proj, double mid_exam, double party = 50);   //constructor
    Student() = default;                                                                //default constructor

    void setID(long id);                                                                //set student's id
    long getID();                                                                       //get student's id
    void setHomework(double hw);                                                        //set student's homeworks grade
    double getHomework();                                                               //get student's homework grade
    void setMidtermProject(double project);                                             //set student's midterm project
    double getMidtermProject();                                                         //get student's midterm project
    void setMidtermExam(double exam);                                                   //set student's midterm exam
    double getMidtermExam();                                                            //get student's midterm exam
    void setParty(double party);                                                        //set amount of party
    double getParty();                                                                  //get amount of party
    double value();                                                                     //likelihood of a student to take the exam
    void show();                                                                        //display informations of a student
                                                                                        //operator
    friend std::ostream& operator<<(std::ostream& os, Student& student);                //display informations of a student
};
```

member function های class Student :

۱)constructor : شامل ۵ ورودی است

- id : شماره دانشجویی
- hw : نمره تمرین ها
- mid_proj : نمره پروژه میانی
- mid_exam : نمره امتحان میان ترم
- party: میزان پارتی با TA که به صورت پیش فرض برابر ۵۰ است.

این constructor به صورت زیر پیاده‌سازی می‌شود.

```
Student::Student(long id, double hw, double mid_proj, double mid_exam, double party)   //constructor
{
    this->id = id;                          //save student's id
    homework = hw;                          //save student's homeworks grade
    midterm_project = mid_proj;             //save student's midterm project grade
    midterm_exam = mid_exam;                //save student's midterm exam grade
    this->party = party;                    //save amont of party
}
```

۲)default constructor : به صورت خودکار تعریف و همه المان ها را برابر صفر قرار می‌دهد.

۳)getID : شماره دانشجویی را برمی‌گرداند و به صورت زیر پیاده‌سازی می‌شود.

```
long Student::getID()                   //get student's id
{
    return id;
}
```

۴)setID : شماره دانشجویی جدید را به عنوان ورودی می‌گیرد و جایگزین قبلی می‌کند.

```
void Student::setID(long id)             //set student's id
{
    this->id = id;
}
```

۵)setHomework : نمره جدید تمرین را از ورودی می‌گیرد و جایگزین قبلی می‌کند.

```
void Student::setHomework(double hw)      //set student's homeworks grade
{
    homework = hw;
}
```

۶)getHomework : نمره تمرین را برمی‌گرداند و به صورت زیر پیاده‌سازی می‌شود.

```
double Student::getHomework()             //get student's homework grade
{
    return homework;
}
```

۷)setmidtermProject : نمره جدید پروژه را از ورودی گرفته و جایگزین قبلی می‌کند.

```
void Student::setMidtermProject(double project)//set student's midterm project
{
    midterm_project = project;
}
```

۸)getmidtermProject : نمره پروژه میانی را برمی‌گرداند و به صورت زیر پیاده‌سازی‌شود.

```
double Student::getMidtermProject()      //get student's midterm project
{
    return midterm_project;
}
```

۹)setMidtermExam : نمره جدید امتحان میان‌ترم از ورودی می‌گیرد و جایگزین قبلی می‌کند.

```
void Student::setMidtermExam(double exam)//set student's midterm exam
{
    midterm_exam = exam;
}
```

۱۰)getmidtermProject : نمره امتحان میان‌ترم را برمی‌گرداند و به صورت زیر پیاده‌سازی می‌شود.

```
double Student::getMidtermExam()       //get student's midterm exam
{
    return midterm_exam;
}
```

۱۱)setParty : میزان جدید پارتی دانشجو با TA را از ورودی می‌گیرد و جاگزین قبلی می‌کند.

```
void Student::setParty(double party)   //set amount of party
{
    this->party = party;
}
```

۱۲)getparty : میزان پارتی دانشجو با TA را برمی‌گرداند و به صورت زیر پیاده‌سازی می‌شود.

```
double Student::getParty()             //get amount of party
{
    return party;
}
```

۱۳)value : این معیاری است که به وسیله این در data structure دانشجوها را مرتب می‌کنیم و الگوریتم محاسبه آن به صورت زیر است.

$$\text{value} = -\frac{(\text{homework} + \text{midterm\_project} + \text{midterm\_exam})}{3} + \text{party}$$

این تابع به صورت زیر استفاده می‌شود.

```
double Student::value()            //likelihood of a student to take the exam
{
    return -(homework + midterm_project + midterm_exam) / 3 + party; //calculate and return value
}
```

۱۴)show : اطلاعات هر دانشجو را نمایش می‌دهد و به صورت زیر نمایش می‌دهد.

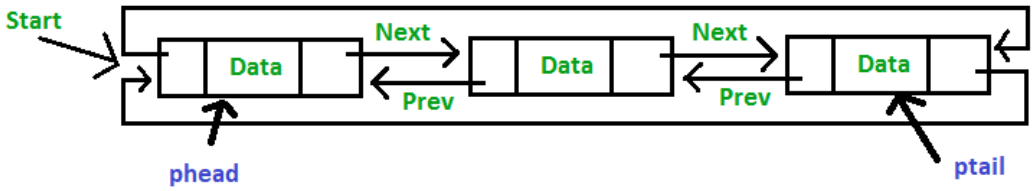
```
void Student::show()                //display informations of a student
{
    std::cout << "Student ID:" << std::setw(10) << id << std::setw(20) << "Homework:" << std::setw(5) << homework;
    std::cout << std::setw(20) << "Midterm project:" << std::setw(5) << midterm_project << std::setw(17);
    std::cout << "Midterm exam:" << std::setw(5) << midterm_exam << std::setw(10) << "Party:" << std::setw(5) << party;
    std::cout << std::setw(10) << "Value:" << std::setw(5) << value() << std::endl;
}
```

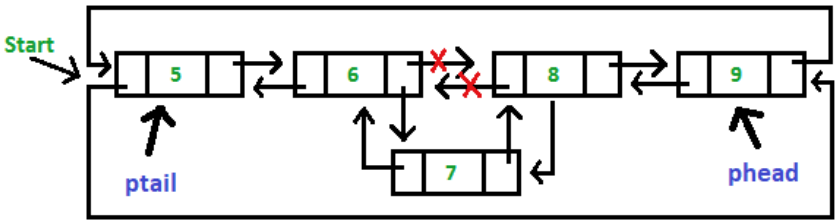
۱۵)operator<< : کار این operator چاپ کردن بعضی ویژگی های دانشجو است. قابل توجه اینکه خروجی ostream است و prototype را به صورتfriend و در داخل خود classتعریف می‌شود که باعث می شود به متغیر های private اجازه دسترسی شود و به صورت زیر پیاده سازی می‌شود .

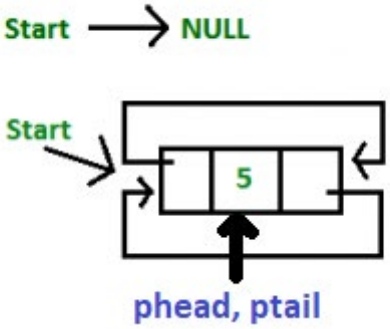
```
std::ostream& operator<<(std::ostream& os, Student& student)  //define operator <<
{                                                             //display informations of a student
    os << "Student ID:" << std::setw(10) << student.id << std::setw(20) << "Homework:" << std::setw(5) << student.homework;
    os << std::setw(20) << "Midterm project:" << std::setw(5) << student.midterm_project << std::setw(17);
    os << "Midterm exam:" << std::setw(5) << student.midterm_exam << std::setw(10) << "Party:" << std::setw(5) << student.party;
    os << std::setw(10) << "Value:" << std::setw(5) << student.value() << std::endl;

    return os;
}
```

دومین class APDS است ؛ که در آن data structure به روش priority Doubly Circular Linked List پیاده‌سازی می‌شود . در شکل های زیر الگوریتم این روش نمایش داده شده.







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 APDS دسترسی پیدا کنیم باید آن‌ها را به صورت public تعریف کرد.

member variable های class Item :

- ps : یک متغییر از جنس pointer بوده که در آن آدرس object student در آن ذخیره می‌شود.
- pnext : یک متغییر از جنس pointer بوده که در آن آدرس object Item بعدی در آن ذخیره می‌شود.
- pprev : یک متغییر از جنس pointer بوده که در آن آدرس object Item قب در آن ذخیره می‌شود.

member function های class Item :

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

پیاده‌سازی این class به صورت زیر است.

```
APDS::Item::Item(Student& S)
{
    ps = &S;
    pnext = nullptr;        //initialize pnext
    pprev = nullptr;        //initialize pprev
}
```

جواب Qusetion1 : اگر ورودی به صورت pass by value بود چون در اینجا می‌خواهیم آدرس object student را ذخیره کنیم ؛ این متغیر چون local است بعد از تمام شدن تابع حذف می‌شود پس ذخیره کردن آدرس آن عملا کار بیهوده‌ایي است ولی اگر ورودی به صورت reference به آن داده شود این مشکل به وجود نمی‌آید چون object student دیگر یک متغییر local نیست که بعد از اتمام تابع از بین برود.

۲)phead : یک متغییر از جنس pointer بوده که در آن آدرس object Item ابتدایی(head) (دانشجویی که بیشترین مقدار value را داخل data structure دارد) در آن ذخیره می‌شود.

۳)ptail : یک متغییر از جنس pointer بوده که در آن آدرس object Item انتهایی(tail) (دانشجویی که کمترین مقدار value را داخل data structure دارد) در آن ذخیره می‌شود.

۴)size : تعداد دانشجویان داخل data structure .

۵)capacity : حداکثر تعداد پذیرش دانشجو در داخل data structure.

۶)name : اسم data structure

در زیرکل class APDS را مشاهده می‌کنیم.

```
class APDS
{
private:                    //member variable class APDS
    class Item              //class Item
    {
    public:
        Student* ps;        //current student
        Item* pnext;        //adress next item
        Item* pprev;        //adress previos item
        Item(Student&);     //constructor class Item
    };

    Item* phead;            //adress of the student with the maximum value in the data structure
    Item* ptail;            //adress of the student with the minimum value in the data structure
    size_t size;            //number of students in data struture
    size_t capacity;        //capacity of our data structure
    std::string name{ "AP Data Structure" };  //name

public:                                                 //member function class APDS

    APDS(std::vector<Student>& students, size_t capacity = 3);   //constructor
    APDS(std::vector<Student>&& students, size_t capacity = 3);  //constructor for rvalue
    APDS(const APDS&);                                  //copy constructor
    APDS();                                             //default costructor
    ~APDS();                                            //destructor

    bool push(Student&);                                //gets a student and tries to push it to the data structure if possible
    bool pop();                                         //removes the student with the least value (tail)
    Student& go(int n);                                 //go n step in data structure
    Student* searchByID(long id);                       //searches for a particular student in the data structure
    void setCapacity(size_t i);                         //changes the capacity
    size_t getCapacity();                               //get capacity
    void setName(std::string s) const;                  //changes the data structure name
    size_t getSize();                                   //get size
    void show() const;                                  //display our data structure
                                                        //operators
    Student& operator[](int n) const;                   //operator []
    APDS& operator=(const APDS& A);                     //operator =
    friend std::ostream& operator<<(std::ostream& os, APDS& input); //display our data structure
};
```

member function های class APDS :

۱)constructor که دارای دو ورودی است.

- students : وکتوری از object student است.
- capacity : حداکثر تعداد پذیرش دانشجو در داخل data structure که به صورت پیش‌فرض برابر ۳ است.

در اینجا constructor هر دانشجو از وکتور ورودی را با استفاده از تابع push در data structure ذخیره می‌کند و با توجه به ظرفیت ممکن است که دانشجویان با value کم را نگیرد.
پیاده‌سازی این تابع به صورت زیر است.

```
APDS::APDS(std::vector<Student>& students, size_t capacity)   //constructor
{
    phead = nullptr;            //initialize phead
    ptail = nullptr;            //initialize ptail
    size = 0;                   //initialize size
    this->capacity = capacity;  //save capacity
    for (auto & x : students)
        push(x);
}
```

۲)copy constructor : ورودی آن یک object از class APDS است که به صورت const و reference به آن داده شده است . چون ما در class APDS هر دفعه برای ذخیره item ها آن را new کردیم پس باید هم copy constructor,operator=,destructor را خودمان تعریف کنیم.
پیاده‌سازی این تابع به صورت زیر است.

```
APDS::APDS(const APDS& A)
{
    capacity = A.capacity;   //save capacity
    size = A.size;           //save size
    name = A.name;           //save name

    phead = new Item{ *(A.phead->ps) };   //save item phead
    Item* new_item{ phead };
    Item* temp = A.phead;

    while (temp->pnext != A.ptail)        //copy all item in data structure
    {
        temp = temp->pnext;
        new_item->pnext = new Item{ *(temp->ps) };
        new_item->pnext->pprev = new_item;
        new_item->pprev = temp->pprev;
        new_item = new_item->pnext;
    }
    new_item->pnext = new Item{ *(A.ptail->ps) };   //save ptail
    new_item->pnext->pprev = new_item;
    ptail = new_item->pnext;
    phead->pprev = ptail;
    ptail->pnext = phead;
}
```

۳)default constructor که در آن member variable ها مقدار دهی اولیه می‌شوند فقط مقدار capacity را برابر ۳ قرار می‌دهیم. پیاده‌سازی این تابع به صورت زیر است.

```
APDS::APDS()                //default constructor
{
    phead = nullptr;        //initialize phead
    ptail = nullptr;        //initialize ptail
    size = 0;               //initialize size
    capacity = 3;           //default capacity

}
```

۴)destructor : این تابع حافظه‌هایی که با new کردن از سیستم گرفته بودیم ، هرجا لازم باشد آن را حذف و به سیستم برمی‌گرداند. این تابع به صورت زیر پیاده‌سازی می‌شود.

```
APDS::~APDS()  //destructor
{              //remove all item in data structure
    Item* temp{ phead->pnext };

    for (size_t i{}; i < size-1; i++)
    {
        delete temp->pprev;
        temp = temp->pnext;
    }
    delete ptail;
}
```

۵)push : ورودی آن یک object student است . کار این تابع سعی در گنجاندن این student در داخل data structure است. اگر data structure ظرفیت پذیرش داشته باشد از روی student یک object Item می‌سازد سپس باید جای مناسب برای این دانشجو پیدا شود. چیدمان در داخل data structure براساس value است ؛ بعد از یافتن جای مناسب حال باید آدرس item بعدی و قبلی را در داخل item جدید ذخیره کرده و آدرس های قبلی را به روز رسانی کرد(در شکل های صفحه توضیح داده شده). اگر data structure ظرفیت پذیرش نداشته باشد ابتدا برسی می‌کنیم که آیا value دانشجو ورودی از value ، ptail کمتر است یا نه ؛ اگر بود که آن دانشجو را نمی‌گیریم ولی اگر نبود با call کردن تابع pop آخرین item که همان ptail است را حذف و دانشجو جدید را به عنوان ورودی می‌گیریم. اگر تابع موفق به گرفتن دانشجو ورودی شود true را برمی‌گرداند در غیر این صورت مقدار false را برمی‌گرداند. پیاده‌سازی این تابع به صورت زیر است.

```
bool APDS::push(Student& S)     //gets a student and tries to push it to the data structure if possible
{
    if (!(capacity - size) && (ptail->ps->value() > S.value()))      //if this value of this student less than ptail return false
        return false;
    else if (!(capacity - size) && (ptail->ps->value() < S.value())) //if this value of this student bigger than ptail ,remove ptail and get the new student
        pop();                                                       //removes the student with the least value (tail)

    Item* new_s =new Item{S} ;    //create new item with this new student

    if (size == 0)                //for save first data
    {
        new_s->pnext = new_s;     //save adresses
        new_s->pprev = new_s;
        phead = new_s;            //phead and ptail for size 1 is same
        ptail = new_s;
        size++;

        return true;
    }

    Item* temp{ phead };     //strat with phead
    size_t counter{ size };  //for save size befor change it

    for (size_t i{}; i < counter; i++)      //search for a good pleace
    {
        if (S.value() > temp->ps->value())  //if find a good place
        {
            temp->pprev->pnext = new_s;     //save this item
            new_s->pprev = temp->pprev;     //save new adresses and update old adrsses
            new_s->pnext = temp;
            temp->pprev = new_s;
            if (i == 0)                     //if phead change
                phead = new_s;
            size++;                         //increase size data structure

            return true;
        }
        temp = temp->pnext;               //go to next student
    }

    //if there is no good place so save this data as new ptail
    new_s->pnext = phead;         //save new adresses and update old adrsses
    new_s->pprev = temp->pprev;
    temp->pprev->pnext = new_s;
    phead->pprev = new_s;
    ptail = new_s;                //save new ptail
    size++;                       //increase size data structure

    return true;
}
```

۶)pop : وقتی این تابع call شود در صورتی که عضوی در data structure وجود داشته باشد ؛ item با کمترین مقدار value را از داخل data structure حذف میکند و size یکی کم میکند در صورتی که عضوی نباشد false را به عنوان خروجی برمی‌گرداند و اگر حذف کند true را برمی‌گرداند. پیاده‌سازی این تابع به صورت زیر است.

```
bool APDS::pop()     //removes the student with the least value (tail)
{
    if(!size)        //If the data structure is empty return false
        return false;

    //removes last data and update adress
    ptail->pprev->pnext = phead;
    phead->pprev = ptail->pprev;
    delete ptail;           //delete ptail
    ptail = phead->pprev;

    size--;                 //decrease size data structure

    return true;
}
```

۷)go : کار این تابع پیمایش data structure است که تعداد گام‌های پیمایش را در ورودی به آن می‌دهیم . باید توجه کرد که پیماپش از head(n=0) شروع می‌شود و در صورتی که n مثبت باشد رو به جلو و در صورت منفی بودن n رو به عقب پیمایش می‌کند. پیاده‌سازی این تابع به صورت زیر است.

```
Student& APDS::go(int n)  //go n step in data structure
{
    if (n >= 0)                   //positive n
    {
        Item* temp{ phead };      //first student
        for (int i{}; i < n; i++) //go  next student
            temp = temp->pnext;

        return *(temp->ps);       //return derefrence student*
    }
    else                          //negitive n
    {
        Item* temp{ phead };      //first student

        for (int i{}; i > n; i--) //go pervious student
            temp = temp->pprev;

        return *(temp->ps);      //return derefrence student*
    }
}
```

۸)searchByID : این تابع یک شماره دانشجویی را به عنوان ورودی می‌گیرد و سپس در داخل data structure شروع به جستجو می‌کندکه دانشجو با این شماره دانشجویی را پیدا کند و در صورت وجود محل ذخیره آن برمیگرداند و در صورت عدم وجود nullptr برمی‌گرداند. پیاده‌سازی این تابع به صورت زیر است.

```
Student* APDS::searchByID(long id)//searches for a particular student in the data structure
{
    Item* temp{ phead };              // start from first student

    for (size_t i{}; i < size; i++)   //search for id
    {
        if (!(temp->ps->getID() - id)) //check is that student or not
            return temp->ps;
        temp = temp->pnext;           //go to next student
    }
    return nullptr;                   //if there is no student with this id return nullptr
}
```

۹) setCapacity : این تابع ظرفیت جدید را از ورودی می‌گیرد و آن را ذخیره می‌کند . اگر ظرفیت جدید کمتر بود و باید فضا برای برخی data ها نباشد باید data هایی که کمترین value را حذف کنیم تا تعداد data ها با ظرفیت برابر شود. این تابع به صورت زیر پیاده‌سازی می‌شود.

```
void APDS::setCapacity(size_t i)   //changes the capacity
{
    capacity = i;
    int def{ static_cast<int>(size - i) };//number of student which we must remove

    if (def > 0)
        for (int j{}; j < def; j++)
            pop();                //removes the student with the least value (tail)
}
```

۱۰)getCapacity : ظرفیت data structure را برمی‌گرداند و به صورت زیر پیاده‌ساز می‌شود.

```
size_t APDS::getCapacity()  //get capacity
{
    return capacity;
}
```

۱۱)getSize : تعداد data های ما را برمی‌گرداند و به صورت زیر پیاده‌سازی می‌شود.

```
size_t APDS::getSize()  //get size
{
    return size;
}
```

۱۲)show : کار این تابع نمایش data structure است .

```
void APDS::show() const //display our data structure
{
    std::cout << "Name: " << name << std::endl;
    std::cout << "Capacity: " << capacity << std::endl;
    std::cout << "Size: " << size << std::endl;
    std::cout << std::setw(20) << "N" << std::setw(20) << "ID" << std::setw(20) << "Homework" << std::setw(20);
    std::cout << "MidProject" << std::setw(20) << "MidExam" << std::setw(20) << "Party" << std::setw(20);
    std::cout << "Value" << std::endl;

    Item* temp{ phead };      //first student

    for (size_t i{}; i < size; i++)
    {
        std::cout << std::setw(20) << i + 1 << std::setw(20) << temp->ps->getID() << std::setw(20) << temp->ps->getHomework();
        std::cout << std::setw(20) << temp->ps->getMidtermProject() << std::setw(20) << temp->ps->getMidtermExam();
        std::cout << std::setw(20) << temp->ps->getParty() << std::setw(20) << temp->ps->value() << std::endl;
        temp = temp->pnext;
    }

}
```

جواب Qusetion2 : برای اجرا شدن کد زیر باید دو تغییر ایجادکرد اول اینکه باید چون object APDS به صورت const تعریف شده فقط method های const را می‌توان call کرد پس باید show را به صورت const تعریف کرد دومین تغییر این است باید constructor ایی را تعریف کنیم که ورودی آن یک وکتور از Student ها باشدکه به صورت rvalue باشد و در داخل خود constructor کل ورودی را در یک متغییر static نگه داریم تا بعد از اتمام برنامه حذف نشود (نمی‌شود به متغییر rvalue ، را به صورت reference به ورودی دهیم). این constructor جدید به صورت زیر تعریف می‌شود.

```cpp
APDS::APDS(std::vector<Student>&& students, size_t capacity) //constructor for rvalue
{
    static std::vector<Student> stu{ students };  //to save students vector

    phead = nullptr;                              //initialize phead
    ptail = nullptr;                              //initialize ptail
    size = 0;                                     //initialize size
    this->capacity = capacity;                    //save capacity
    for (auto &x : stu)
        push(x);
}
```

۱۳)setName : اسم جدید data structure را از ورودی گرفته و ذخیره می‌کند .

جواب Qusetion3 : برای اجرا شدن کد زیر باید دو تغییر ایجادکرد اول اینکه باید چون object APDS به صورت const تعریف شده فقط method های const را می‌توان call کرد پس باید setName را به صورت const تعریف کرد و چون ما در این تابع داریم member variable را تغییر می‌دهیم ابتدا باید object را non const کرد سپس می‌شود اسم جدید را ذخیره کرد . تابع جدید به صورت زیر پیاده‌سازی می‌شود.

```cpp
void APDS::setName(std::string s) const     //changes the data structure name to s
{
    auto p = const_cast<APDS*>(this);
    p->name = s;
}
```

۱۴)operator[] : ورودی این operator شماره data ها از صفر تا size-1 است که اولین phead و آخری ptail است. اگر ورودی مثبت باشد به سمت جلو و اگر منفی باشد رو به عقب پیمایش می کند و reference دانشجو مورد نظر را برمی‌گرداند و در صورتی که ورودی خارج از محدوده بود reference یک student که همه اطلاعات آن صفر است را برمی‌گرداند. این تابع به صورت زیر پیاده‌سازی می‌شود.

```cpp
student& APDS::operator[](int n) const    //operator []
{
    if (n >= 0 && n < size)               //positive n
    {
        Item* temp{ phead };              //first student
        for (int i{}; i < n; i++)         //go  next student
            temp = temp->pnext;

        return *(temp->ps);               //return derefrence student*
    }
    else if (n < 0 && n >= -static_cast<int>(size))      //negitive n
    {
        Item* temp{ phead };       //first student

        for (int i{}; i > n; i--)  //go pervious student
            temp = temp->pprev;

        return *(temp->ps);        //return derefrence student*
    }
    else                           //out of range n
    {
        static Student new_student{};

        return new_student;        //reference to new student which has zero information
    }
}
```

۱۵)operator= : چون ما در class APDS هر دفعه برای ذخیره item ها آن را new کردیم پس باید هم operator= را خودمان تعریف کنیم. الگوریتم آن ، حذف کردن اطلاعات قبلی وکپی کردن اطلاعات جدید است. پیاده‌سازی این تابع به صورت زیر است.

```
APDS& APDS::operator=(const APDS& A)  //operator =
{
    Item* temp = nullptr;

    if (size != 0)
    {
        temp = phead->pnext;         //remove all item in data structure
        for (size_t i{}; i < size - 1; i++)
        {
            delete temp->pprev;
            temp = temp->pnext;
        }
        delete ptail;
    }

    capacity = A.capacity;   //save capacity
    size = A.size;           //save size
    name = A.name;           //save name

    phead = new Item{ *(A.phead->ps) };   //save item phead
    Item* new_item{ phead };
    temp = A.phead;
    while (temp->pnext != A.ptail)        //copy all item in data structure
    {
        temp = temp->pnext;
        new_item->pnext = new Item{ *(temp->ps) };
        new_item->pnext->pprev = new_item;
        new_item->pprev = temp->pprev;
        new_item = new_item->pnext;
    }
    new_item->pnext = new Item{ *(A.ptail->ps) };   //save ptail
    new_item->pnext->pprev = new_item;
    ptail = new_item->pnext;
    phead->pprev = ptail;
    ptail->pnext = phead;

    return *this;
}
```

۱۶)operator<< : کار این operator نمایش data structure است. قابل توجه اینکه خروجی ostream است و prototype را به صورتfriend و در داخل خود classتعریف می‌شود که باعث می شود به متغیر های private اجازه دسترسی شود و به صورت زیر پیاده سازی می‌شود .

```
std::ostream& operator<<(std::ostream& os, APDS& input)      //display our data structure
{
    os << "Name: " << input.name << std::endl;
    os << "Capacity: " << input.capacity << std::endl;
    os << "Size: " << input.size << std::endl;
    os << std::setw(20) << "N" << std::setw(20) << "ID" << std::setw(20) << "Homework" << std::setw(20);
    os << "MidProject" << std::setw(20) << "MidExam" << std::setw(20) << "Party" << std::setw(20);
    os << "Value" << std::endl;

    APDS::Item* temp{ input.phead }; //first student

    for (size_t i{}; i < input.size; i++)
    {
        os << std::setw(20) << i + 1 << std::setw(20) << temp->ps->getID() << std::setw(20) << temp->ps->getHomework();
        os << std::setw(20) << temp->ps->getMidtermProject() << std::setw(20) << temp->ps->getMidtermExam();
        os << std::setw(20) << temp->ps->getParty() << std::setw(20) << temp->ps->value() << std::endl;
        temp = temp->pnext;
    }

    return os;   //return out put
}
```

۱۷)getHeadstudent : این تابع دانشجویی که به عنوان head است را برمی‌گرداند.

```
Student* APDS::getHeadStudent() //get phead
{
    return phead->ps;
}
```

بعد از اتمام تعریف class ها چند تابع دیگر لازم است تعریف شود . prototype های آن ها را در زیر مشاهده می‌کنید.

```
std::vector<Student> getData(const char* filename);  //read .csv file and get students
void show(std::vector<Student> student);             //show students
void show(const APDS& ap);                           //shows properties of an APDS object
```

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

    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
        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;

        for (size_t i = 0; i < student.length(); i++)  //to get data from a line
        {
            if (student[i] != ',')
                data += student[i];
            else
            {
                if (j == 0)         //set id
                    input_student.setID(static_cast<long>(std::stod(data)));
                else if (j == 1)    //set homework
                    input_student.setHomework(std::stod(data));
                else if (j == 2)    //set midterm project
                    input_student.setMidtermProject(std::stod(data));
                else if (j == 3)    //set midterm exam
                    input_student.setMidtermExam(std::stod(data));

                data = "";
                j++;
            }
        }
        input_student.setParty(std::stod(data)); //set party
        output.push_back(input_student);        //save student
    }
    input_file.close();

    return output;
}
```

۲)show : ورودی این تابع وکتوری از student ها می‌باشد و این تابع باید اطاعات هر دانشجو را نمایش بدهد و به صورت زیر پیاده‌سازی می‌شود.

```cpp
void show(std::vector<Student> student)     //show students
{
    std::cout << std::setw(20) << "N" << std::setw(20) << "ID" << std::setw(20) << "Homework" << std::setw(20);
    std::cout << "MidProject" << std::setw(20) << "MidExam" << std::setw(20) << "Party" << std::setw(20);
    std::cout << "Value" << std::endl;

    for (size_t i = 0; i < student.size(); i++)
    {
        std::cout << std::setw(20) << i + 1 << std::setw(20) << student[i].getID() << std::setw(20) << student[i].getHomework();
        std::cout << std::setw(20) << student[i].getMidtermProject() << std::setw(20) << student[i].getMidtermExam();
        std::cout << std::setw(20) << student[i].getParty() << std::setw(20) << student[i].value() << std::endl;
    }
}
```

۳)show : ورودی این تابع یک object از class APDS است که به صورت const و reference به آن داده شده است وکار این تابع نشان داده اطلاعات data structure است. در واقع همان method show است که در class APDS پیاده‌سازی کردیم و به صورت زیر پیاده‌سازی می‌شود.

```cpp
void show(const APDS& ap)       //shows properties of an APDS object
{
    ap.show();
}
```

جواب Qusetion4 : ۱)برای دسترسی به متغییر های private در یک باید prototype تابع را در خود class مدنظر قرار داد و قبل از آن کلمه friend را بگذاریم که باعث می‌شود در داخل تابع اجازه دسترسی به متغییرهای private داده می‌شود . مانند operator>> در بالا.

۲)راه دوم این که از طریق آدرس متغییر های private به آن دسترسی پیدا کنیم. مانند زیرکه سعی شده تابع show آخری را با استفاده از این روش پیاده سازی کرد.

```cpp
void show(const APDS& ap)       //shows properties of an APDS object
{

    size_t* adress = (size_t*)&ap;
    size_t size{ *adress };
    size_t capacity{ *(adress + 1) };


    std::cout << "Capacity: " << capacity << std::endl;
    std::cout << "Size: " << size << std::endl;
    std::cout << std::setw(20) << "N" << std::setw(20) << "ID" << std::setw(20) << "Homework" << std::setw(20);
    std::cout << "MidProject" << std::setw(20) << "MidExam" << std::setw(20) << "Party" << std::setw(20);
    std::cout << "Value" << std::endl;

    for (int i{}; i < size; i++)
    {
        std::cout << std::setw(20) << i + 1 << std::setw(20) << ap[i].getID() << std::setw(20) << ap[i].getHomework();
        std::cout << std::setw(20) << ap[i].getMidtermProject() << std::setw(20) << ap[i].getMidtermExam();
        std::cout << std::setw(20) << ap[i].getParty() << std::setw(20) << ap[i].value() << std::endl;
    }


}
```

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.

توجه۳: این پروژه همانطورکه در تصاویر مشخص است ابتدا در visual studio 2019 نوشته و سپس در visual studio code به وسیله docker اجرا و نتایج زیر بدست آمده است.

نتایج google test به شرح زیر است.

```
RUNNING TESTS ...
[==========] Running 3 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 3 tests from APHW4Test
[ RUN      ] APHW4Test.StudentTest
Student ID:   9423013          Homework:   80    Midterm project:   95    Midterm exam:   90    Party:   50    Value:-38.3333
[       OK ] APHW4Test.StudentTest (1 ms)
[ RUN      ] APHW4Test.APDSTest
[       OK ] APHW4Test.APDSTest (0 ms)
[ RUN      ] APHW4Test.dataTest
[       OK ] APHW4Test.dataTest (0 ms)
[----------] 3 tests from APHW4Test (3 ms total)

[----------] Global test environment tear-down
[==========] 3 tests from 1 test suite ran. (6 ms total)
[  PASSED  ] 3 tests.
Here!
<<<SUCCESS>>>
```

همانطورکه قابل مشاهده است نتایج google test موفقیت آمیز بوده.

تمام