0600603151

המסלול האקדמי המכללה למינהל

ביה"ס למדעי המחשב

המסלול האקדמי
המכללה למינהל

ברקוד נבחן	ת.ז הסטודנט: _____ מספר חדר: _____ מספר נבחן: _____ מספר אסמכתא: _____

מבחן בקורס: תכנות מונחה עצמים

תאריך הבחינה: 27.09.13

שנת הלימודים: תשע"ג, **סמסטר:** קיץ, **מועד:** א'

משך הבחינה: 3.5 שעות

שם המרצה/ים:
אליהו חלסצ'י

שם המתרגל/ים:
חיים שפיר

מבנה הבחינה: הבחינה מורכבת מחלק אחד.

מספר השאלות הכולל בבחינה: 6.

משקל כל שאלה: בצמוד לכל שאלה

הוראות לנבחן:

- אסור השימוש בכל חומר עזר
- יש לענות בגוף השאלון.
- נדרש להחזיר את השאלון.
- לא מצורף נספח לבחינה
- מחברת טיוטה: כן
- מחברת נפרדת לכל שאלה: לא
-

בהצלחה!!

שאלה 1 – Constructors and virtual methods (10 נקודות)

נתונות ההגדרות של המחלקות הבאות :

```
class A{
   public:
   A(){
      cout<<"A"<<endl;
      m();    // notice!!
   }
   void call_m(){
      //...
      m();
   }
   virtual void m(){
      cout<<"A::m()"<<endl;
   }
   ~A(){
      cout<<"~A"<<endl;
      m();    // notice!!
   }
};

class B: public A{
   public:
   B(){
      cout<<"B"<<endl;
   }
   virtual void m(){
      cout<<"B::m()"<<endl;
   }
   ~B(){
      cout<<"~B"<<endl;
   }
};

int main()
{
   B b;
   b.call_m();
   return 0;
}
```

מהו הפלט של התוכנית? (10 נק')

תשובה:

```
A
A::m()
B
B::m()
~B
~A
A::m()
```

שאלה 2: ירושה ופולימורפיזם (30 נקודות)

אנו בונים סימולציה של רובוטים. נתונה המחלקה הבאה המגדירה רובוט, מחלקה זו מגדירה את כל המשותף לכל הרובטים השונים בסימולציה, כמו למשל המיקום שלהן במפה הנקבע ע"י x,y:

```
class Robot{
   …
  public:
  Robot(int x,int y){ …  }
   …
};
```

מחלקה זו ניתנה לנו בקוד סגור – היא אינה ניתנת לעריכה ולא ניתן להניח קיומן של מתודות שונות פרט למה שמופיע לעיל.

בסימולציה קיימים רובוטים שונים. הם מורכבים מחלקים שונים כמו מנשא חפצים, זרוע, חיישנים שונים וכו'.

התחלנו לבנות את המחלקה הבאה המתארת מנשא שבתוכו הרובוט יכול לשמור חפצים שאסף:

```
class Carrier{
  int m_load;        // הקיבולת המקסימלית של המנשא
  char** m_items_names;          // שמות החפצים שאסף
  int num_of_items;        // מספר החפצים שאסף
  public:
  Carrier (const char** a_items_names, int a_load){
   // העתקת מערך שמות החפצים וכמותם
    m_load=a_load;
  }
   int getLoad() const {return m_load;}
   …
};
```

<u>הערה</u>: אין להוסיף בונה דיפולטיבי למחלקה Carrier.

א. כיתבו copy Ctor עבור המחלקה Carrier (4 נק')

ב. כיתבו (בצורה מונחית עצמים) את המחלקה MyCarrierRobot ואת הבונה שלה, ע"פ ההגדרות הבאות: (10 נק')

a. MyCarrierRobot הוא סוג של רובוט,

b. שיש לו מנשא.

c. על הבונה לקבל פרמטרים שבאמצעותם יש לאתחל את MyCarierRobot.

ג. נתון מערך של רובוטים **Robot המאותחל בכל מיני סוגים שונים של רובוטים.

כתבו פונקציה (גלובאלית) המקבלת כפרמטרים את המערך, את גודלו של המערך, ופרמטר קיבולת. על הפונקציה להחזיר את מספר המנשאים השייכים לרובוטים מסוג MyCarrierRobot שהקיבולת שלהם גדולה מפרמטר הקיבולת (16 נק')

לדוגמא אם פרמטר הקיבולת היה 50, והרובוטים מסוג MyCarrierRobot שבמערך שלנו מחזיקים מנשאים עם הקיבולות הבאות: 40,50,50,100,100,80,30,20,10 אז הפונקציה שלנו צריכה להחזיר 3.

דגשים:

* פרט לפונקציה, ניתן להוסיף מתודות רק למחלקה MyCarrierRobot

* הקפידו על תכנות מונחה עצמים נכון – לדוגמא, מתודה לא תחזיר ערך שלא שייך ישירות למחלקה שלה

* הקפידו על יעילות – כיצד פרמטרים מועברים

* שימו לב שלא כל הרובוטים במערך הם מסוג MyCarrierRobot

פתרון:

א.

```
Carrier(const Carrier & a_carrier){
 num_of_items=a_carrier.num_of_items;
 m_items_names=new char*[num_of_items];
 for(int i=0;i<num_of_items;i++){
        m_items_names[i]=new char[strlen(a_carrier.m_items_names[i])+1];
        strcpy(m_items_names[i], a_carrier.m_items_names[i]);
 }
  m_load=a_carrier.m_load;
}
```

ב.

```
class MyCarrierRobot : public Robot{
   Carrier carrier;
   public:
   MyCarrierRobot (int x,int y,const Carrier& c) :Robot(x,y) , carrier(c) { }
};
```

ג.

למחלקה MyCarrierRobot נוסיף את המתודה הבאה:

```
const Carrier& MyCarrierRobot::getCarrier(){ return carrier; }
```

ונכתוב את הפונקציה:

```
int count_loadss(Robot** arr, int size, int load){
    int count=0;
    for(int i=0;i<size;i++){
            if(typeid(arr[i]) == typeid(MyCarrierRobot*)) {      // if it is a carrier robot
                    MyCarrierRobot* r = (MyCarrierRobot*)arr[i];      // convert to warrior
                    if( (r->getCarrier()).getLoad() > load)
                            count++
            }
    }
    return count;
}
```

שאלה 3: Operator Overloading – Const (20 נקודות)

נתונה המחלקה הבאה המכילה מערך של int-ים

```
class MyArray{
   int* arr;
   int size;
    // ...
}
```

כמו כן, נתונה פונקציית ה main הבאה:

```
int main() {
   MyArray a(3);  // create an inner array of 3 ints
   a[0]=5;
   a[1]=6;
   a[2]=7;
   cout<<a<<"---"<<endl;
  return 0;
}
```

פלט:

```
5
6
7
---
```

השלימו את הקוד החסר כך שפונקציית ה main תעבוד ללא בעיות. שימו לב שה main לא בודקת את כל מקרי הקצה, ואתם נדרשים להקפיד על יעילות, צורת העברת הפרמטרים השונים (בדגש על const) וערכי החזרה.

תשובה:

```
class MyArray{
   int* arr;
   int size;
   public:
   MyArray(int a_size){
      size=a_size;
      arr=new int[size];
   }
   int getSize() const { return size;}
   int& operator[](int index){return arr[index];}
   int operator[] (int index) const {return arr[index];}
};

ostream& operator<<(ostream& out,const MyArray& a){
   for(int i=0;i<a.getSize();i++){
      out<<a[i]<<endl;
   }
   return out;
}
```

שאלה 4 – Templetes (20 נקודות)

נתונה פונקציית ה main הבאה:

```
int main() {
   int i=2;
   float f=2.2;
   Dub dub;            // notice!!!
   dub(i);
   dub(f);
   cout << i << endl;  // 4
   cout << f << endl;  // 4.4
  return 0;
}
```

למרות ש i ו f הינם טיפוסים שונים, הפעולה dub מכפילה את ערכם

א. ממשו את הנדרש כדי שהקוד ב main יעבוד. (10 נק')

ב. כתבו פונקציה בשם Apply הפועלת על מבנה נתונים לא ידוע (יכול להיות מערך, רשימה מקושרת, עץ וכו'), המכיל נתונים מסוג לא ידוע (מכיל למשל סטודנטים או עובדים או int'ים וכו') ובנוסף הפונקציה תקבל כפרמטר פעולה כלשהי. הפונקציה Apply תפעיל את הפעולה הנ"ל על כל אחד מהנתונים במבנה הנתונים. (10 נק')

תשובה:

א.

```
class Dub{
   public:
   template <class T>
   void operator()(T& t){
      t*=2;
   }
};
```

ב.

```
template<class iterator, class operation>

void Apply(iterator begin, iterator end, const operation& f) {

for( ; begin != end ; begin++)

   f(*begin);

}
```

שאלה 5 שאלות פתוחות (10 נק')

א. בשאלה 4 בסעיף א', מהו Dub מהו dub? (4 נק')

ב. בשאלה 4 בסעיף ב', מהם ההנחות הסמויות שהנחת על הפרמטרים השונים? (4 נק')

ג. מה הקשר בין Dub ל Apply? (2 נק')

תשובות:

א. Dub היא מחלקה המגדירה object function – מחלקה עם אופרטור () הפועל על טיפוס T כלשהו ומכפיל את ערכו פי 2. dub הוא אובייקט שנוצר ממחלק הזו ובאמצעותו ניתן להפעיל את האופרטור ().

ב. שלאובייקט מסוג iterator מומשו האופרטורים =! , ++ , *, ושניתן לקבל אותם by value בבטיחות. ולאובייקט מסוג operation קיים האופרטור () הפועל על טיפוס פרמטרי T.

ג. אובייקט מסוג Dub ניתן להעברה כפרמטר הפעולה עבור apply, אנו מעבירים "פונקציה" כפרמטר אבל בפעול פשוט מעבירים כפרמטר אובייקט של Dub.

שאלה 6: ירושה מרובה (10 נקודות)

האם קטע הקוד הבא מתקמפל, אם לא – מדוע?

אם כן – מה מודפס למסך?

```
class Shape{
   char color;
   public:
   Shape(char c){
      color=c;
   }
   // ...
};

class Rectangle : virtual public Shape{
   int width,height;
   public:
   Rectangle(char c):Shape(c){};
   //...

};
class Circle : virtual public Shape{
   int x,y,r;
   public:
   Circle(char c):Shape(c){};
   //...
};

class Cylinder : public Rectangle, public Circle{
   public:
   Cylinder(char c):Rectangle(c),Circle(c){};
};

int main()
{
 Cylinder c(1);
}
```

תשובה : הקוד לא מתקמפל.

מדוע? מכיוון שה CTOR של המחלקה Cylinder מורכב תחילה מ Shape ואז מהשאר, ולכן הוא דורש אתחול של Shape ואז את האתחול של השאר. לא ניתן ליצור Rectangle שיאתחל את Shape (גם אם מדובר באותו ה shape של circle) כי לא ניתן ליצור את האובייקט בשכבה השנייה לפני שהאובייקטו בשכבה הראשונה נוצר...

הדרך הנכונה ביותר היתה לבקש ב CTOR שני אובייקטים, האחד מסוג Rectangle והשני מסוג Circle, ובאמצעות ה Rectangle למשל לאתחל תחילה את Shape ואז את שתי הצורות האחרות בהתאמה.