CONFIDENTIAL

UTM UNIVERSITI TEKNOLOGI MALAYSIA SCHOOL OF COMPUTING Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAMINATION (PRACTICAL)

SEMESTER II 2018/2019

SUBJECT CODE	**: SCSJ1023**
SUBJECT NAME	**: PROGRAMMING TECHNIQUE II**

SOLUTIONS

Question 1 **[35 marks]**

Line | **Corrected Program**

```
1    //Program 1
2    class Food
3    {
4       string desc;
5       double price;
6
7       public:
8          Food(string desc, double price) {
9            this->desc = desc;
10           this->price = price;
11         }
12
13         string getDesc() const { return desc; } //(2M)
14         double calcPriceInRinggit() const { return price * USTOMYR; }
15         virtual void displayInfo() { //(2M)
16           cout << fixed << setprecision(2)
17                << "Price: USD" << price << endl
18                << "Price converted to Malaysian = MYR"
19                << calcPriceInRinggit() << endl << endl; //(2M)
20         }
21   };
22
23   class Vegetable : public Food //(2M)
24   {
25      int weight;
26
27      public:
28         Vegetable(string desc, double price, int weight)
29         : Food(desc, price) { //(4M)
30           this->weight = weight;
31         }
32
33         double calcWeightInGram() const {
34           return weight * POUNDTOGRAM;
35         }
36
37         ~~virtual~~ void displayInfo() { //(2M)
38           cout << "Food description: " << getDesc() << endl
39                << "Weight in pound: " << weight << " pound" << endl
40                << "Weight in gram: " << calcWeightInGram() //(2M)
41                << " grams" << endl;
42           Food::displayInfo(); //(2.5M)
43         }
44   };
45
46   class CannedFood : public Food //(2M)
47   {
48      string type, expDate;
49
```

```
   public:
      CannedFood(string desc, double price, string type, string
      expDate): Food(desc, price) { //(4M)
        this->type = type;
        this->expDate = expDate;
      }

      ~~virtual~~ void displayInfo() { ///(2M)
        cout << "Food description: " << getDesc() << endl
             << "Canned Food Type: " << type << endl
             << "Expired date: " << expDate << endl;
        Food::displayInfo(); //(2.5M)
      }
};

int main()
{
   Food *f[] = { new Vegetable("Broccoli", 1.6, 3),
                 new Vegetable("Tomato", 1.4, 5),
                 new CannedFood("Mushroom Soup", 5.78, "Soups",
                                "12/09/2020"),
                 new Vegetable("Cabbage", 0.7, 4.5),
                 new CannedFood("Sliced Yellow Cling Peaches",
                                9.58, "Fruit", "01/02/2021")};

   for (int i = 0; i < sizeof(f) / sizeof(f[0]); i++) //(4M)
   {
      cout << "Food #" << (i + 1) << endl;
      f[i]->displayInfo(); //(2M)
   }

   return 0;
}
```

Question 2 **[65 marks]**

Task	Answer (C++ Statements)
1 **(5M)**	`StoreData::StoreData() { /* set id, names, and sales data to 0*/ id = 0; name[0] = 0; for (int i = 0; i < MAX_MONTH; i++) sales[i] = 0; }`
2 **(2M)**	`int StoreData::getId() const //accessors for id { return id; }`

3 **(2M)**

```
char* StoreData::getName() const   //accessors for name
{
      return (char *)name;
}
```

4 **(2M)**

```
float* StoreData::getSales() const  //accessors for sales
{
      return (float*)sales;
}
```

5 **(2M)**

```
void StoreData::setCounter(int c)   //mutators for _counter
{
      _counter = c;
}
```

6 **(3M)**

```
void StoreData::setName(char name[])      //mutators for name
{
      for (int i = 0; i < MAX_STORE_NAME; i++)
            this->name[i] = name[i];
}
```

7 **(3M)**

```
void StoreData::setSales(float sales[])   //mutators for sales
{
      for (int i = 0; i < MAX_MONTH; i++)
            this->sales[i] = sales[i];
}
```

8 **(5M)**

```
ostream& operator<<(ostream& os, const StoreData& sd)
{
      //supply stream with: id, name and sales data

      os << "[" << sd.id << "]" <<"\t"<< sd.name<<"\t";

      for (int i = 0; i < MAX_MONTH; i++)
            os << " " << sd.sales[i];

      return os;
}
```

9 **(3M)**

```
StoreManager::~StoreManager()
{
      if (storedata != 0)     //if there's allocated data
      {
            //free allocated memory
            delete[] storedata;
      }
}
```

10 **(9M)**

(a) **(2M)**

```
temp = new StoreData [store_data_count + 1];
```

(b) **(2M)**

```
for (int i = 0; i < store_data_count; i++)
   temp[i] = storedata[i];
```

(c) **(2M)**

```
temp[store_data_count] = s;
```

(d) **(1M)**

```
store_data_count++;
```

(e) **(2M)**

```
storedata = new StoreData[1];
```

11 **(7M)**

```
/* search for data with matching id */
for (int i = 0; i < store_data_count; i++)            //1
{
```

	`if (storedata[i].getId() == id)                //1`
	`{`
	`storedata[i].setName(s.getName());         //2`
	`storedata[i].setSales(s.getSales());       //2`
	
	`/* exit function */`
	`return;                                    //1`
	`}`
	`}`

12	**(5M)**

(a) **(3M)**	`if (store_data_count == 0)`
	`{`
	`cout << " No data to print ! " << endl;`
	`return;`
	`}`

(b) **(2M)**	`for (int i = 0; i < store_data_count; i++)`
	`cout << storedata[i] << endl;`

13 **(2M)**	`StoreData* StoreManager::getStoreData() const`
	`{`
	`return storedata;`
	`}`

14	**(5M)**
(a) **(1M)**	`ofstream fc(filename.data(), ios::binary);`
(b) **(3M)**	`fc.write((char *)s.getStoreData(), s.getStoreDataLength());`
(c) **(1M)**	`fc.close();`
15	**(10M)**
(a) **(1M)**	`ifstream fc(filename.data(), ios::binary);`

(b) **(2M)**	`if (!fc)`
	`{`
	`cout << "Error !!! file not found : " << filename << endl;`
	`return;`
	`}`

(c) **(2M)**	`fc.seekg(0L, ios::end);`
	`file_length = fc.tellg();`

(d) **(1M)**	`count = file_length / sizeof(StoreData);`
(e) **(1M)**	`temp = new StoreData[count];`

(f) **(2M)**	`fc.seekg(0L);`
	`fc.read((char *)temp, file_length);`

(g) **(1M)**	`fc.close();`