5 חלק יבש

5.1 סעיף א

```
template <class T>
std::vector<T> slice(std::vector<T> vec, int start, int step, int stop);
```

כתבו פונקציה בשם slice בעלת החתימה המופיעה מעלה. הפונקציה מחזירה וקטור חדש המכיל את כל הערכים מ-vec החל מאינדקס start (כולל) ועד אינדקס stop (לא כולל), בקפיצות של step. לדוגמה:

```
// this syntax initializes a vector with values a,b,c,d,e
std::vector<char> vec1 {'a', 'b', 'c', 'd', 'e'};
// returns vector with values a,c
std::vector<char> vec_sliced = slice(vec1, 0, 2, 4);
// returns vector with values b,c,d,e
std::vector<char> vec_sliced = slice(vec1, 1, 1, 5);
```

הערות:

- אם start קטן מ-0 או גדול או שווה לגודל הוקטור, זרקו חריגת BadInput.
- אם stop קטן מ-0 או גדול מגודל הוקטור, זרקו חריגת BadInput.
- אם step קטן או שווה ל-0, זרקו חריגת BadInput.
- אם start גדול או שווה ל-stop, אז הוקטור המוחזר יהיה ריק (אלא אם כן הדרישות האחרות מחייבות זריקת חריגה).

פתרון:

```
template<class T>
std::vector<T> slice(std::vector<T> vec, int start, int step, int stop) {
    if (start < 0 || start ≥ vec.size() || stop < 0 || stop > vec.size() || step ≤ 0) {
        throw BadInput();
    }
    std::vector<T> new_vec = {};
    if (start ≥ stop) {
        return new_vec;
    }
    for (int i = start; i < stop; i += step) {
        new_vec.push_back(vec[i]);
    }
    return new_vec;
}
```

5.2 סעיף ב

נתון קטע הקוד הבא:

```
class A {
public:
  std::vector<int*> values;
  void add(int x) { values.push_back(new int(x)); }
};

int main() {
  A a, sliced;
  a.add(0); a.add(1); a.add(2); a.add(3); a.add(4); a.add(5);
  sliced.values = slice(a.values, 1, 1, 4);
  *(sliced.values[0]) = 800;
  std::cout << *(a.values[1]) << std::endl;
  return 0;
}
```

כתבו גרסה חדשה למחלקה A, על מנת שהרצת ה-main תענה על הדרישות הבאות:

1. הפונקציה תדפיס למסך את המספר 800.
2. לא יתרחשו דליפות זיכרון, גישה לזיכרון לא מאותחל, שחרור כפול או שגיאות זיכרון אחרות.

הערות:

- הניחו שהפונקציה slice קיימת, ושהיא פועלת לפי המתואר בסעיף א'.
- מותר לשנות אך ורק את המחלקה A. אסור לשנות את ה-main.
- אין צורך לציין include-ים.

פתרון:

```
class A {
public:
    std::vector<std::shared_ptr<int>> values;

    void add(int x) {
        std::shared_ptr<int> value_pointer(new int(x));
        values.push_back(value_pointer);
    }
};
```