भारतीय सूचना प्रौद्योगिकी संस्थान गुवाहाटी

Indian Institute of Information Technology Guwahati

DATA STRUCTURES LAB (CS111)

ASSIGNMENTS–03

1. Create an array of integers with dynamic memory allocation. The size of the array is user input. Read the elements of the array as inputs. Write a function to perform insertion sort on that array.

2. Consider the following structure:

```
struct student_t {
    char *name;
    int roll;
    float cpi;
};
```

   Create a dynamically allocated array of size $n = 3$ for storing pointers to objects of type `struct student_t`. Let a pointer `p` point to the allocated memory location. Let $m = 0$ be the initial number of elements in the array. Create a menu-based program to perform the following operations. You need to write separate functions for each operation. The details about the students are user inputs.

   i. *Insert an Element at the End:* If you have space $(m < n)$ in the array, insert an element at the $(m-1)$th location. Increase m by one. Else create an array of size $\lceil 1.2 \times n \rceil$. Then, copy all elements of the previous array to the new array and delete (deallocate) the old array. p now points to the new array. Now, insert an element at the new array's $(m-1)$th location. Increase m by one. Update n.

   ii. *Insert an Element at the Beginning:* Use a similar approach as explained earlier. However, in this case, you must shift all elements in the array to the right by an element to make space at the 0th location.

   iii. *Insert an Element at the ith Location:* Use a similar approach as explained earlier. However, in this case, you must shift all elements in the array between index $(i+1)$ and $(m-0)$ to the right by an element to make space at the 0th location. If $i > m-1$, insert it at the $(m-1)$th location.

iv. *Delete an Element from the End:* You must deallocate the memory for the object and decrease m by one. After deleting the element, if $m < 0.5 \times n$, create an array of size $\lceil 0.75 \times n \rceil$. Then copy all elements to the new array. Delete the old array. Set $m = \lceil 0.75 \times n \rceil$. Update n.

v. *Delete an Element from the Beginning:* Use a similar scheme as described earlier. However, you must shift all the elements to the left by an element.

vi. *Delete an Element from the ith Location:* Use a similar scheme as described earlier. Shift the required elements.

vii. *Sort by* `roll`: Use insertion sort.

viii. *Sort by* `cpi`: Use insertion sort.

Figure 1 shows an example of the array's memory layout. Use the following code to sort the array:

```
struct student_t {
    char *name;
    int roll;
    float cpi;
};

typedef struct student_t *Student;
typedef int (*Comparator)(Student, Student);

int compareRoll(Student a, Student b) {
    return a->roll > b->roll ? 1 : 0;
}

int compareCPI(Student a, Student b) {
   return a->cpi > b->cpi ? 1 : 0;
}

void sortStudents(Student *p, int m, Comparator c) {
    for (int i = 1; i < m; i++) {
        Student key = p[i];
        int j = i - 1;
        while (j >= 0 && c(p[j], key)) {
            p[j + 1] = p[j];
            j = j - 1;
        }
        p[j + 1] = key;
    }
}

int main() {
    int n = 2, m = 0;
    Student *p = calloc(n, sizeof (Student));
    // Write code here
```

```
    sortStudents(p, m, compareRoll);
    // Write code here
    sortStudents(p, m, compareCPI);
    // Write code here
    return 0;
}
```

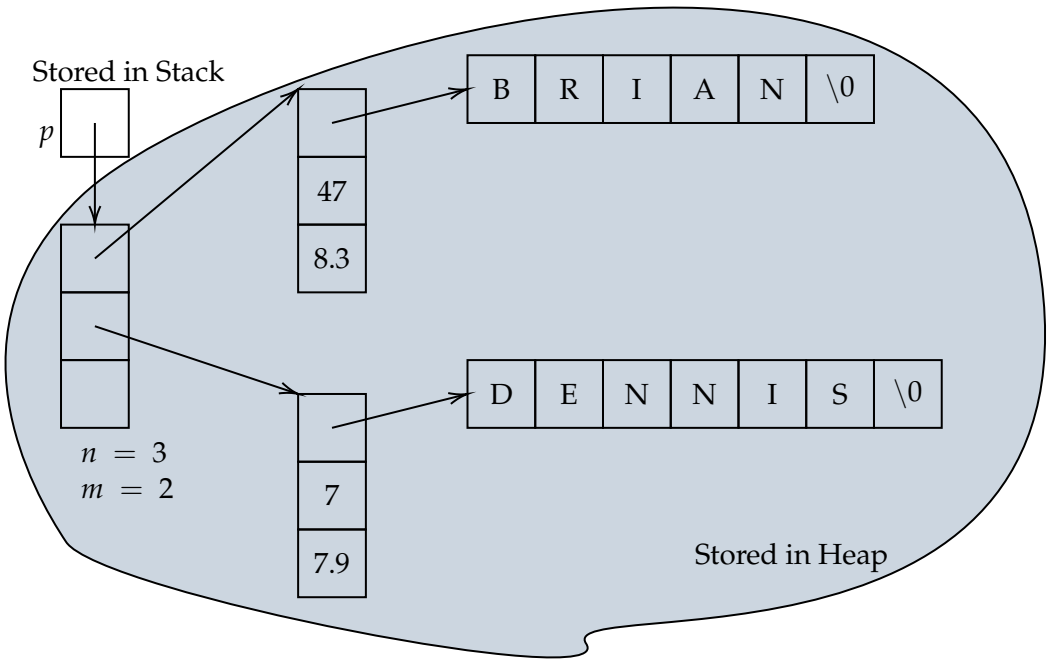

Figure 1: An example of the memory layout of the array.