

# Data Structure Assignment - 1

## Problems

- Write a C program to perform the following operations using an array:
  - Insert element into array
  - Delete element from array
  - Search a given element using linear search technique
  - Search a given element using binary search technique
- Write a C program for sparse matrix representation
- Write a C program to add two polynomial using arrays

## Problem - 1

### C Code

```

1  /**
2  * Data Structures Assignment - 1
3  * CS 392
4  * Problem 1:
5  * - Write a C program to perform the following operations using an array:
6  *   - Insert element into array
7  *   - Delete element from array
8  *   - Search a given element using linear search technique
9  *   - Search a given element using binary search technique
10 *
11 * Joydeep Mukherjee
12 * CSE,3rd Sem, 11000116030
13 * GCETTS 2017
14 *
15 **/
16
17 #include <stdio.h>
18 #include <stdlib.h>
19
20 void show (int *arr,int len); // Show the array
21 void menu (int **arr,int len); // menu
22
23 int input (int len); // Takes input & bound checks
24
25 int create (int **arr); // Input array from user & return the size of array
26 int insert (int **arr,int len); // Insert element
27 int delete (int **arr,int len); //Delete element
28
29 void linSearch (int *arr,int len); // linear search
30 void binSearch (int *arr,int len); // binary search
31 int binSearchRun (int *arr,int s,int l,int h); // Run binary Search
32 int SelectionSort (int *arr,int len); // Selection Sort
33
34 int main(int argc, char const *argv[]) {
35     /* Lets define the program parameters
36     - An array of int

```

```
37     - An int containing length of array
38     */
39
40     int *array = NULL;
41     int len = 0;
42
43     len = create(&array); // Lets input our array from stdin
44     menu(&array,len);    // Show our array
45
46     return 0;
47 }
48
49 int create(int **arr) {
50     /* Prompts user to enter the array
51        Arguments: a pointer to a pointer
52        Returns: the number of items in array
53        Summary: Updates the pointer to point to a memory location on
54               created on heap, containing the array of elements
55     */
56
57     int* temp_array = NULL;
58     int i,len = 0;
59
60     /* Input the length of array */
61     printf("Create a new array\n");
62     printf("Number of array elements: ");
63     scanf("%d",&len);
64
65     temp_array = (int*) malloc(sizeof(int)*len); // Create the array
66
67     /* Input the array */
68     printf("Enter the elements: ");
69     for (i = 0; i < len; i++) {
70         scanf("%d",(temp_array + i));
71     }
72
73     *arr = temp_array; // Set the main pointer address
74     fflush(stdin);     // Sometimes prevents extra input to be passed to menu
75
76     return len;
77 }
78
79 void show(int *arr,int len) {
80     /* Shows the array
81     Arguments: the array,length
82     Returns: None
83     */
84
85     int i;
86
87     printf("\nThe array is:\n");
88
89     for (i = 0; i < len-1; i++)
90         printf("%d, ",*(arr + i));
91     printf("%d\n",*(arr + i));
92
93 }
94
95 void menu(int **arr,int len) {
96     /*
97     Prompts user to input a choice according to the list
```

```
98     Arguments: a pointer to a pointer
99     Returns: Nothing
100 */
101
102 int sel = -1;
103 fflush(stdin);
104 printf("Choose an option:\n");
105 printf("\t1> Show the array \n");
106 printf("\t2> Insert element \n");
107 printf("\t3> Delete element \n");
108 printf("\t4> Search element with linear search\n");
109 printf("\t5> Search element with binary search\n");
110 printf("\t6> Sort the array with Selection Sort\n");
111 printf("\t7> Re-enter array\n");
112 printf("\t0> Exit\n");
113
114 do {
115     /* menu options */
116     printf("\nYour choice:");
117     fflush(stdin);
118     sel = input(8);
119
120     /* Call Functions accordingly */
121     switch (sel) {
122         case 1:
123             show(*arr, len);
124             break;
125         case 2:
126             len = insert(arr, len);
127             break;
128         case 3:
129             len = delete(arr, len);
130             break;
131         case 4:
132             linSearch(*arr, len);
133             break;
134         case 5:
135             binSearch(*arr, len);
136             break;
137         case 6:
138             SelectionSort(*arr, len);
139             break;
140         case 7:
141             len = create(arr);
142             break;
143     }
144 } while(sel);
145
146 return;
148 }
149
150 int input(int len) {
151     /* Takes input from user in the range [0, len-1]
152     Arguments: a limit len
153     Returns: a number from user in range
154     */
155
156     int temp = -1;
157     while (1) {
158         fflush(stdin);
```

```
159     scanf("%d",&temp);
160
161     if (temp < 0 || temp >= len)
162         printf("[Out of range] Retry:");
163     else
164         break;
165 }
166
167 return temp;
168 }
169
170 int insert(int **arr,int len) {
171     /* Inserts an element at a specific position
172     Arguments: A pointer to a pointer, initial lenght
173     Returns: Updated lenght
174     Summary: Reads index & value from stdin
175             Creates a new array & copies values from old to new
176             frees the old, changes the pointer
177     */
178
179     int i,index,val;
180     int *temp_array = (int*) malloc(sizeof(int)*(len+1));
181
182     printf("\nInsert an element in the zero-indexed array\n");
183     printf("\tEnter Index: ");
184     index = input(len);
185     printf("\tEnter Value: ");
186     scanf("%d",&val);
187
188     /* Iterate & copy elements before our desired index */
189     for( i = 0; i < index; i++)
190         *(temp_array + i) = (*(arr) + i);
191
192     *(temp_array + i) = val; // Insert new element
193     /* Copy remaining elements upto end */
194     for(; i < len; i++)
195         *(temp_array + i + 1) = (*(arr) + i);
196
197     free(*arr); // Free the previous block, prevents memory leak
198     *arr = temp_array; // Update our pointer
199
200     return len+1;
201 }
202
203 int delete(int **arr,int len) {
204     /* Deletes an element from a specific position
205     Arguments: The pointer to the array pointer,lenght
206     Returns: The modified length
207     Summary: Reads index & value from stdin
208             Creates a new array & copies values from old to new
209             frees the old, changes the pointer
210     */
211
212     int index,i;
213     int *temp_array = (int*) malloc(sizeof(int)*(len-1));
214
215     printf("\nDelete an element from the zero-indexed array\n");
216     printf("\tEnter index:");
217     index = input(len);
218
219     for( i = index + 1;i < len;i++)
```

```
220     (*arr + i - 1) = (*arr + i);
221
222     *arr = realloc(*arr,sizeof(int)*(len - 1)); // Removes extra space
223     return len-1;
224 }
225
226 // linear search
227 void linSearch(int *arr,int len) {
228     /* Does a linear search on the array
229     Arguments: a pointer to an array, length of array
230     Returns: None
231     */
232
233     int s,i;
234     printf("\nSearch an element in the array using LinearSearch\n");
235     printf("\tEnter value:");
236     scanf("%d",&s);
237
238     for (i = 0; i < len; i++) {
239         if (*(arr + i) == s) {
240             printf("\tFound %d at index %d\n",s,i);
241             return;
242         }
243     }
244
245     printf("\tFound %d at NO index\n",s);
246 }
247
248 // binary search
249 void binSearch(int *arr,int len) {
250     /* Does a binary search on the array for a specific value taken from stdin
251     Arguments: a pointer to an array
252     Returns: None
253     Calls: SelectionSort()
254     */
255
256     int s,i;
257
258     SelectionSort(arr,len);
259
260     printf("\nSearch an element in the array using BinarySearch\n");
261     printf("\tEnter value:");
262     scanf("%d",&s);
263
264     i = binSearchRun(arr,s,0,len);
265
266     if (i != -1)
267         printf("\tFound %d at index %d\n",s,i);
268     else
269         printf("\tFound %d at NO index\n",s);
270 }
271
272 // binary search
273 int binSearchRun(int *arr,int s,int l,int h) {
274     /* Searches an element in the array using binary search
275     Arguments: a pointer to an array, target variable,lower end,higher end
276     Returns: index of target variable, if none then -1
277     */
278
279     int mid = (l + h)/2;
280
```

```

281     if(h < l)
282         return -1;
283     else if (s == *(arr + mid))
284         return mid;
285     else if (s > *(arr + mid))
286         return binSearchRun(arr,s,mid+1,h);
287     else if (s < *(arr + mid))
288         return binSearchRun(arr,s,l,mid-1);
289 }
290
291 // Selection Sort
292 int SelectionSort(int *arr,int len) {
293     /* Sorts the array using Selection Sort algorithm
294     Arguments: pointer to the array,length of array
295     Returns: 0
296     */
297
298     int i,j,vi,vj;
299     for (i = 0; i < len; i++) {
300         for (j = 0; j < len; j++) {
301
302             vi = *(arr + i);
303             vj = *(arr + j);
304
305             if (vi < vj) {
306                 *(arr + i) = vj;
307                 *(arr + j) = vi;
308             }
309         }
310     }
311     printf("\nArray is sorted using Selection Sort\n");
312     return 0;
313 }
314

```

[https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1\\_1.c](https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1_1.c)

## Output:

```

1 Create a new array
2 Number of array elements: 3
3 Enter the elements: 34 51 68
4 Choose an option:
5 1> Show the array
6 2> Insert element
7 3> Delete element
8 4> Search element with linear search
9 5> Search element with binary search
10 6> Sort the array with Selection Sort
11 7> Re-enter array
12 0> Exit
13
14 Your choice:2
15
16 Insert an element in the zero-indexed array
17 Enter Index: 3
18 [Out of range] Retry:2
19 Enter Value: 85
20

```

```
21 Your choice:2
22
23 Insert an element in the zero-indexed array
24 Enter Index: 0
25 Enter Value: 17
26
27 Your choice:1
28
29 The array is:
30 17, 34, 51, 85, 68
31
32 Your choice:2
33
34 Insert an element in the zero-indexed array
35 Enter Index: 0
36 Enter Value: 170
37
38 Your choice:2
39
40 Insert an element in the zero-indexed array
41 Enter Index: 0
42 Enter Value: 153
43
44 Your choice:2
45
46 Insert an element in the zero-indexed array
47 Enter Index: 4
48 Enter Value: 135
49
50 Your choice:1
51
52 The array is:
53 153, 170, 17, 34, 135, 51, 85, 68
54
55 Your choice:4
56
57 Search an element in the array using LinearSearch
58 Enter value:34
59 Found 34 at index 3
60
61 Your choice:4
62
63 Search an element in the array using LinearSearch
64 Enter value:35
65 Found 35 at NO index
66
67 Your choice:5
68
69 Array is sorted using Selection Sort
70
71 Search an element in the array using BinarySearch
72 Enter value:17
73 Found 17 at index 0
74
75 Your choice:1
76
77 The array is:
78 17, 34, 51, 68, 85, 135, 153, 170
79
80 Your choice:86
81 [Out of range] Retry:5
```

```
82
83 Array is sorted using Selection Sort
84
85 Search an element in the array using BinarySearch
86 Enter value:86
87 Found 86 at NO index
88
89 Your choice:3
90
91 Delete an element from the zero-indexed array
92 Enter index:3
93
94 Your choice:5
95
96 Array is sorted using Selection Sort
97
98 Search an element in the array using BinarySearch
99 Enter value:51
100 Found 51 at index 2
101
102 Your choice:1
103
104 The array is:
105 17, 34, 51, 85, 135, 153, 170
106
107 Your choice:3
108
109 Delete an element from the zero-indexed array
110 Enter index:6
111
112 Your choice:5
113
114 Array is sorted using Selection Sort
115
116 Search an element in the array using BinarySearch
117 Enter value:170
118 Found 170 at index 6
119
120 Your choice:1
121
122 The array is:
123 17, 34, 51, 85, 135, 153
124
125 Your choice:3
126
127 Delete an element from the zero-indexed array
128 Enter index:4
129
130 Your choice:1
131
132 The array is:
133 17, 34, 51, 85, 153
134
135 Your choice:4
136
137 Search an element in the array using LinearSearch
138 Enter value:135
139 Found 135 at NO index
140
141 Your choice:0
142
```



## Problem - 2

### C Code

```
1  /**
2  * Data Structures Assignment - 1
3  * CS 392
4  * Problem 2:
5  * - Write a C program for sparse matrix representation
6  *
7  *
8  * Joydeep Mukherjee
9  * CSE,3rd Sem, 11000116030
10 * GCETTS 2017
11 *
12 **/
13
14 #include <stdio.h>
15
16 int main(int argc, char const *argv[]) {
17
18     int r,c,i,j,size;
19     int arr[100][100];
20
21     printf("Enter Row Size: ");
22     scanf("%d",&r);
23     printf("Enter Column Size: ");
24     scanf("%d",&c);
25
26     for (i = 0; i < r; i++)
27         for (j = 0; j < c; j++)
28             scanf("%d",(*(arr + i) + j) );
29
30     printf("Sparse matrix of %dx%d is:\n",r,c);
31     for (i = 0; i < r; i++) {
32         for (j = 0; j < c; j++) {
33             printf("%d ",(*(arr + i) + j) );
34         }
35         printf("\n");
36     }
37
38     for (i = 0; i < r; i++)
39         for (j = 0; j < c; j++)
40             if (arr[i][j] != 0)
41                 size++;
42
43     printf("Size of matrix:%d\n",size);
44     int compactMatrix[3][size];
45
46     // Making of new matrix
47     int k;
48
49     for (i = 0; i < 4; i++)
50         for (j = 0; j < 5; j++)
51             if (arr[i][j] != 0)
52                 {
53                     compactMatrix[0][k] = i;
54                     compactMatrix[1][k] = j;
55                     compactMatrix[2][k] = arr[i][j];
```

```
56         k++;
57     }
58
59     printf("Sparse matrix:\n");
60     for (int i=0; i<3; i++)
61     {
62         for (int j=0; j<size; j++)
63             printf("%d ", compactMatrix[i][j]);
64         printf("\n");
65     }
66     return 0;
67 }
```

[https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1\\_2.c](https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1_2.c)

## Output

```
1 Enter Row Size: 3
2 Enter Column Size: 4
3 0 0 1 2
4 3 4 0 0
5 0 5 6 0
6 Sparse matrix of 3x4 is:
7 0 0 1 2
8 3 4 0 0
9 0 5 6 0
10 Size of matrix:6
11 Sparse matrix:
12 0 0 1 1 2 2
13 2 3 0 1 1 2
14 1 2 3 4 5 6
```

## Problem - 3

### C Code

```
1 /**
2  * Data Structures Assignment - 1
3  * CS 392
4  * Problem 2:
5  * - Write a C program to add two polynomial using arrays
6  *
7  *
8  * Joydeep Mukherjee
9  * CSE,3rd Sem, 11000116030
10 * GCETTS 2017
11 *
12 **/
13
14 #include <stdio.h>
15 #include <stdlib.h>
16
17 typedef struct element {
18     int power;
19     int mult;
20 } elem;
21
```

```
22 int main(int argc, char const *argv[]) {
23     elem *pa,*pb,*pc;
24     int len,i,j,k;
25
26     printf("Enter order of polynomial:");
27     scanf("%d",&len);
28
29     pa = (elem*) malloc(sizeof(elem)*len);
30     pb = (elem*) malloc(sizeof(elem)*len);
31     pc = (elem*) malloc(sizeof(elem)*2*len);
32
33     printf("For polynomial:1 enter the multiplier, power in descending order\n");
34     printf("Enter '0 0' to quit\n");
35     printf("Format:[mult][pow]\n");
36     for (i = 0; i < len; i++) {
37         scanf("%d %d",&(pa + i)->mult,&(pa + i)->power);
38         if(!(pa + i)->mult && !(pa + i)->power)
39             break;
40     }
41
42     printf("For polynomial:2 enter the multiplier, power in descending order\n");
43     for (i = 0; i < len; i++) {
44         scanf("%d %d",&(pb + i)->mult,&(pb + i)->power);
45         if(!(pb + i)->mult && !(pb + i)->power)
46             break;
47     }
48
49     for (i = 0; i < len*2; i++) {
50         if((pa + j)->power == (pb + k)->power) {
51             (pc + i)->power = (pa + j)->power;
52             (pc + i)->mult = (pa + j++)->mult + (pb + k++)->mult;
53         }
54         else if((pa + j)->power > (pb + k)->power) {
55             (pc + i)->power = (pa + j)->power;
56             (pc + i)->mult = (pa + j++)->mult;
57         }
58         else {
59             (pc + i)->power = (pb + k)->power;
60             (pc + i)->mult = (pb + k++)->mult;
61         }
62     }
63
64     printf("polynomial 1: ");
65     for( i = 0; ((pa + i)->mult && (pa + i)->power) && i < len ; i++)
66         printf("%dx^%d + ",(pa + i)->mult,(pa + i)->power);
67     printf("0\n");
68
69     printf("polynomial 2: ");
70     for( i = 0; ((pb + i)->mult && (pb + i)->power) && i < len ; i++)
71         printf("%dx^%d + ",(pb + i)->mult,(pb + i)->power);
72     printf("0\n");
73
74     printf("Result: ");
75     for( i = 0; ((pc + i)->mult && (pc + i)->power) && i < len*2 ; i++)
76         printf("%dx^%d + ",(pc + i)->mult,(pc + i)->power);
77     printf("0\n");
78
79     return 0;
80 }
81
```

[https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1\\_2.c](https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1_2.c)

## Output

```
1 Enter order of polynomial:5
2 For polynomial:1 enter the multiplier, power in descending order
3 Enter '0 0' to quit
4 Format:[mult][pow]
5 4 5
6 3 2
7 0 0
8 For polynomial:2 enter the multiplier, power in descending order
9 6 6
10 4 5
11
12 3 4
13 1 2
14 2 1
15 polynomial 1:  $4x^5 + 3x^2 + 0$ 
16 polynomial 2:  $6x^6 + 4x^5 + 3x^4 + 1x^2 + 2x^1 + 0$ 
17 Result:  $6x^6 + 8x^5 + 3x^4 + 4x^2 + 2x^1 + 0$ 
```