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

```c
/**
* Data Structures Assignment - 1
* CS 392
* Problem 1:
*  - 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
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/

#include <stdio.h>
#include <stdlib.h>

void show   (int *arr,int len); // Show the array
void menu   (int **arr,int len); // menu

int  input   (int len);  // Takes input & bound checks

int  create (int **arr); // Input array from user & return the size of array
int  insert (int **arr,int len); // Insert element
int  delete (int **arr,int len); //Delete element

void linSearch     (int *arr,int len); // linear search
void binSearch     (int *arr,int len); // binary search
int  binSearchRun  (int *arr,int s,int l,int h); // Run binary Search
int  SelectionSort (int *arr,int len); // Selection Sort

int main(int argc, char const *argv[]) {
  /* Lets define the program parameters
    - An array of int
    - An int containing length of array
  */

  int *array = NULL;
  int len = 0;

  len = create(&array); // Lets input our array from stdin
  menu(&array,len);     // Show our array

  return 0;
}

int create(int **arr) {
  /*  Prompts user to enter the array
      Arguments: a pointer to a pointer
      Returns: the number of items in array
      Summary: Updates the pointer to point to a memory location on
               created on heap, containing the array of elements
  */

  int* temp_array = NULL;
  int i,len = 0;

  /* Input the length of array */
  printf("Create a new array\n");
  printf("Number of array elements: ");
  scanf("%d",&len);

  temp_array = (int*) malloc(sizeof(int)*len);   // Create the array

  /* Input the array */
  printf("Enter the elements: ");
  for (i = 0; i < len; i++) {
    scanf("%d",(temp_array + i));
  }

  *arr = temp_array;   // Set the main pointer address
  fflush(stdin);       // Sometimes prevents extra input to be passed to menu

  return len;
}

void show(int *arr,int len) {
  /* Shows the array
  Arguments: the array,length
  Returns: None
  */

  int i;

  printf("\nThe array is:\n");

  for (i = 0; i < len-1; i++)
    printf("%d, ",*(arr + i));
  printf("%d\n",*(arr + i));

}

void menu(int **arr,int len) {
  /*
    Prompts user to input a choice according to the list
    Arguments: a pointer to a pointer
    Returns: Nothing
  */

  int sel = -1;
  fflush(stdin);
  printf("Choose an option:\n");
  printf("\t1> Show the array \n");
  printf("\t2> Insert element \n");
  printf("\t3> Delete element \n");
  printf("\t4> Search element with linear search\n");
  printf("\t5> Search element with binary search\n");
  printf("\t6> Sort the array with Selection Sort\n");
  printf("\t7> Re-enter array\n");
  printf("\t0> Exit\n");

  do {
    /* menu options */
    printf("\nYour choice:");
    fflush(stdin);
    sel = input(8);

    /* Call Functons accordingly */
    switch (sel) {
      case 1:
        show(*arr,len);
        break;
      case 2:
        len = insert(arr,len);
        break;
      case 3:
        len = delete(arr,len);
        break;
      case 4:
        linSearch(*arr,len);
        break;
      case 5:
        binSearch(*arr,len);
        break;
      case 6:
        SelectionSort(*arr,len);
        break;
      case 7:
        len = create(arr);
        break;
    }

  } while(sel);

  return;
}

int input(int len) {
  /* Takes input from user in the range [0,len-1]
  Arguments: a limit len
  Returns: a number from user in range
  */

  int temp = -1;
  while (1) {
    fflush(stdin);
    scanf("%d",&temp);

    if (temp < 0 || temp >= len)
      printf("[Out of range] Retry:");
    else
      break;
  }

  return temp;
}

int insert(int **arr,int len) {
  /* Inserts an element at a specific position
     Arguments: A pointer to a pointer, initial lenght
     Returns: Updated lenght
     Summary: Reads index & value from stdin
              Creates a new array & copies values from old to new
              frees the old, changes the pointer
  */

  int i,index,val;
  int *temp_array = (int*) malloc(sizeof(int)*(len+1));

  printf("\nInsert an element in the zero-indexed array\n");
  printf("\tEnter Index: ");
  index = input(len);
  printf("\tEnter Value: ");
  scanf("%d",&val);

  /* Iterate & copy elements before our desired index */
  for( i = 0; i < index; i++)
    *(temp_array + i) = *(*(arr) + i);

  *(temp_array + i) = val; // Insert new element
  /* Copy remaining elements upto end */
  for(; i < len; i++)
    *(temp_array + i + 1) = *(*(arr) + i);

  free(*arr);  // Free the previous block, prevents memory leak
  *arr = temp_array; // Update our pointer

  return len+1;
}

int delete(int **arr,int len) {
  /* Deletes an element from a specific position
    Arguments: The pointer to the array pointer,lenght
    Returns:   The modified length
    Summary:   Reads index & value from stdin
               Creates a new array & copies values from old to new
               frees the old, changes the pointer
   */

   int index,i;
   int *temp_array = (int*) malloc(sizeof(int)*(len-1));

   printf("\nDelete an element from the zero-indexed array\n");
   printf("\tEnter index:");
   index = input(len);

   for( i = index + 1;i < len;i++)
```

```
      *(*arr + i -1) = *(*arr + i);

   *arr = realloc(*arr,sizeof(int)*(len - 1)); // Removes extra space
   return len-1;
}

// linear search
void linSearch(int *arr,int len) {
  /* Does a linear search on the array
  Arguments: a pointer to an array, length of array
  Returns: None
  */

  int s,i;
  printf("\nSearch an element in the array using LinearSearch\n");
  printf("\tEnter value:");
  scanf("%d",&s);

  for (i = 0; i < len; i++) {
    if (*(arr + i) == s) {
      printf("\tFound %d at index %d\n",s,i);
      return;
    }
  }

  printf("\tFound %d at NO index\n",s);
}

// binary search
void binSearch(int *arr,int len) {
  /* Does a binary search on the array for a specific value taken from stdin
  Arguments: a pointer to an array
  Returns: None
  Calls: SelectionSort()
  */

  int s,i;

  SelectionSort(arr,len);

  printf("\nSearch an element in the array using BinarySearch\n");
  printf("\tEnter value:");
  scanf("%d",&s);

  i = binSearchRun(arr,s,0,len);

  if (i != -1)
    printf("\tFound %d at index %d\n",s,i);
  else
    printf("\tFound %d at NO index\n",s);
}

// binary search
int binSearchRun(int *arr,int s,int l,int h) {
  /* Searchs an element in the array using binary search
  Arguments: a pointer to an array, target variable,lower end,higher end
  Returns: index of target variable, if none then -1
  */

  int mid = (l + h)/2;

  if(h < l)
    return -1;
  else if (s == *(arr + mid))
    return mid;
  else if (s > *(arr + mid))
    return binSearchRun(arr,s,mid+1,h);
  else if (s < *(arr + mid))
    return binSearchRun(arr,s,l,mid-1);
}

// Selection Sort
int SelectionSort(int *arr,int len) {
  /* Sorts the array using Selection Sort algorithm
  Arguments: pointer to the array,length of array
  Returns: 0
  */

  int i,j,vi,vj;
  for (i = 0; i < len; i++) {
    for (j = 0; j < len; j++) {

      vi = *(arr + i);
      vj = *(arr + j);

      if (vi < vj) {
        *(arr + i) = vj;
        *(arr + j) = vi;
      }
    }
  }
  printf("\nArray is sorted using Selection Sort\n");
  return 0;
}

```

```
https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1_1.c
```

Output:

```
Create a new array
Number of array elements: 3
Enter the elements: 34 51 68
Choose an option:
 1> Show the array
 2> Insert element
 3> Delete element
 4> Search element with linear search
 5> Search element with binary search
 6> Sort the array with Selection Sort
 7> Re-enter array
 0> Exit

Your choice:2

Insert an element in the zero-indexed array
 Enter Index: 3
[Out of range] Retry:2
 Enter Value: 85

```

```
Your choice:2

Insert an element in the zero-indexed array
 Enter Index: 0
 Enter Value: 17

Your choice:1

The array is:
17, 34, 51, 85, 68

Your choice:2

Insert an element in the zero-indexed array
 Enter Index: 0
 Enter Value: 170

Your choice:2

Insert an element in the zero-indexed array
 Enter Index: 0
 Enter Value: 153

Your choice:2

Insert an element in the zero-indexed array
 Enter Index: 4
 Enter Value: 135

Your choice:1

The array is:
153, 170, 17, 34, 135, 51, 85, 68

Your choice:4

Search an element in the array using LinearSearch
 Enter value:34
 Found 34 at index 3

Your choice:4

Search an element in the array using LinearSearch
 Enter value:35
 Found 35 at NO index

Your choice:5

Array is sorted using Selection Sort

Search an element in the array using BinarySearch
 Enter value:17
 Found 17 at index 0

Your choice:1

The array is:
17, 34, 51, 68, 85, 135, 153, 170

Your choice:86
[Out of range] Retry:5
```

```

Array is sorted using Selection Sort

Search an element in the array using BinarySearch
 Enter value:86
 Found 86 at NO index

Your choice:3

Delete an element from the zero-indexed array
 Enter index:3

Your choice:5

Array is sorted using Selection Sort

Search an element in the array using BinarySearch
 Enter value:51
 Found 51 at index 2

Your choice:1

The array is:
17, 34, 51, 85, 135, 153, 170

Your choice:3

Delete an element from the zero-indexed array
 Enter index:6

Your choice:5

Array is sorted using Selection Sort

Search an element in the array using BinarySearch
 Enter value:170
 Found 170 at index 6

Your choice:1

The array is:
17, 34, 51, 85, 135, 153

Your choice:3

Delete an element from the zero-indexed array
 Enter index:4

Your choice:1

The array is:
17, 34, 51, 85, 153

Your choice:4

Search an element in the array using LinearSearch
 Enter value:135
 Found 135 at NO index

Your choice:0

```

Problem - 2

C Code

```c
/**
* Data Structures Assignment - 1
* CS 392
* Problem 2:
*  - Write a C program for sparse matrix representation
*
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/

#include <stdio.h>

int main(int argc, char const *argv[]) {

    int r,c,i,j,size;
    int arr[100][100];

    printf("Enter Row Size: ");
    scanf("%d",&r);
    printf("Enter Column Size: ");
    scanf("%d",&c);

    for (i = 0; i < r; i++)
    for (j = 0; j < c; j++)
      scanf("%d",(*(arr + i) + j) );

    printf("Sparse matrix of %dx%d is:\n",r,c);
    for (i = 0; i < r; i++) {
    for (j = 0; j < c; j++) {
      printf("%d ",*(*(arr + i) + j) );
    }
    printf("\n");
    }

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            if (arr[i][j] != 0)
                size++;

    printf("Size of matrix:%d\n",size);
    int compactMatrix[3][size];

    // Making of new matrix
    int k;

    for (i = 0; i < 4; i++)
        for (j = 0; j < 5; j++)
            if (arr[i][j] != 0)
            {
                compactMatrix[0][k] = i;
                compactMatrix[1][k] = j;
                compactMatrix[2][k] = arr[i][j];
                k++;
            }

    printf("Sparse matrix:\n");
    for (int i=0; i<3; i++)
    {
        for (int j=0; j<size; j++)
            printf("%d ", compactMatrix[i][j]);
        printf("\n");
    }
    return 0;
}
```

```
https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1_2.c
```

Output

```
Enter Row Size: 3
Enter Column Size: 4
0 0 1 2
3 4 0 0
0 5 6 0
Sparse matrix of 3x4 is:
0 0 1 2
3 4 0 0
0 5 6 0
Size of matrix:6
Sparse matrix:
0 0 1 1 2 2
2 3 0 1 1 2
1 2 3 4 5 6
```

Problem - 3

C Code

```
/**
* Data Structures Assignment - 1
* CS 392
* Problem 2:
*  - Write a C program to add two polynomial using arrays
*
*
* Joydeep Mukherjee
* CSE,3rd Sem, 11000116030
* GCETTS 2017
*
**/

#include <stdio.h>
#include <stdlib.h>

typedef struct element {
  int power;
  int mult;
} elem;

```

```c
int main(int argc, char const *argv[]) {
  elem *pa,*pb,*pc;
  int len,i,j,k;

  printf("Enter order of polynomial:");
  scanf("%d",&len);

  pa = (elem*) malloc(sizeof(elem)*len);
  pb = (elem*) malloc(sizeof(elem)*len);
  pc = (elem*) malloc(sizeof(elem)*2*len);

  printf("For polynomial:1 enter the multiplier, power in descending order\n");
  printf("Enter '0 0' to quit\n");
  printf("Format:[mult][pow]\n");
  for (i = 0; i < len; i++) {
    scanf("%d %d",&(pa + i)->mult,&(pa + i)->power);
    if(!(pa + i)->mult && !(pa + i)->power)
      break;
  }

  printf("For polynomial:2 enter the multiplier, power in descending order\n");
  for (i = 0; i < len; i++) {
    scanf("%d %d",&(pb + i)->mult,&(pb + i)->power);
    if(!(pb + i)->mult && !(pb + i)->power)
      break;
  }

  for ( i = 0; i < len*2; i++) {
    if((pa + j)->power == (pb + k)->power) {
      (pc + i)->power = (pa + j)->power;
      (pc + i)->mult = (pa + j++)->mult + (pb + k++)->mult;
    }
    else if((pa + j)->power > (pb + k)->power) {
      (pc + i)->power = (pa + j)->power;
      (pc + i)->mult = (pa + j++)->mult;
    }
    else {
      (pc + i)->power = (pb + k)->power;
      (pc + i)->mult = (pb + k++)->mult;
    }
  }

  printf("polynomial 1: ");
  for( i = 0; ((pa + i)->mult && (pa + i)->power) && i < len ; i++)
    printf("%dx^%d + ",(pa + i)->mult,(pa + i)->power);
  printf("0\n");

  printf("polynomial 2: ");
  for( i = 0; ((pb + i)->mult && (pb + i)->power) && i < len ; i++)
    printf("%dx^%d + ",(pb + i)->mult,(pb + i)->power);
  printf("0\n");

  printf("Result: ");
  for( i = 0; ((pc + i)->mult && (pc + i)->power) && i < len*2 ; i++)
    printf("%dx^%d + ",(pc + i)->mult,(pc + i)->power);
  printf("0\n");

  return 0;
}

```

```
https://github.com/joydeep1701/GCETTS/blob/master/CS392/Assignments/1_2.c
```

Output

```
Enter order of polynomial:5
For polynomial:1 enter the multiplier, power in descending order
Enter '0 0' to quit
Format:[mult][pow]
4 5
3 2
0 0
For polynomial:2 enter the multiplier, power in descending order
6 6
4 5

3 4
1 2
2 1
polynomial 1: 4x^5 + 3x^2 + 0
polynomial 2: 6x^6 + 4x^5 + 3x^4 + 1x^2 + 2x^1 + 0
Result: 6x^6 + 8x^5 + 3x^4 + 4x^2 + 2x^1 + 0
```