Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

WEEK-END ASSIGNMENT-05
Array and Array Processing
Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with arrays for storing and processing collections of values of the same types in different applications.

Assignment Objectives:

To learn how to declare and use arrays for storing collections of values of the same type as well as to learn how to process the elements of an array.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. We initialize a 25-element array with the prime numbers less than 100.

```
int prime_lt_100[] = {2, 3, 5, 7, 11, 13, 17,
    19, 23, 29, 31, 37,41, 43, 47, 53, 59,
   61, 67, 71, 73, 79, 83, 89, 97};
```

Determine the array elements given in the following expressions;

```
(a) prime_lt_100[24];
(b) int i=10; prime_lt_100[i+4];
(c) prime_lt_100[prime_lt_100[2] +
                          prime_lt_100[0]];
(d) prime_lt_100[6]=prime_lt_100[6] +
                          prime_lt_100[16];
```

Output

2. Draw the required array with it's content as per the given code snippet.

```
#define SIZE 6
int sqaure[SIZE],i;
for(i=0;i<SIZE;i++){
      square[i]=i*i;
}
```

Output

3. Consider the following C program; **[GATE 2019]**

```
int main(){
  int arr[]={1,2,3,4,5,6,7,8,9,0,1,2,5};
  int *ip=arr+4;
  printf("%d\n",ip[1]);
  return 0;
}
```

Output▼

4. State the output of the code snippet;

```
int sub();
int add(int);
int main(){
  int i;
  int arr[]={10,20,add(30),sub()+10};
  for(i=0;i<4;i++)
     printf("arr[%d]=%d\n",i,arr[i]);
  return 0;
}
int add(int x){
  return x;
}
```

```
int sub(){
  int i=10;
  return(i+30);
}
```

Output▼

5. Write the output of the code snippet;

```
int main(){
  int arr[]={1,2,3,4,5,6,7,8,9,0};
  for(int i=0;i<10;i++)
      printf("%d ",arr[i]);
  printf("\n");
  for(int i=0;i<10;i++)
      printf("%d ",i[arr]);
  printf("\n");
  for(int i=0;i<10;i++)
      printf("%d ",*(arr+i));
  printf("\n");
  for(int i=0;i<10;i++)
      printf("%d ",*(&arr[i]));
  return 0;
}
```

Output▼

6. Consider the following C program **[GATE 2019]**

```
int main(){
  int a[]={2,4,6,8,10};
  int sum=0,i,*b=a+4;
  for(i=0;i<5;i++){
     sum=sum+(*b - i) - *(b - i);
  }
  printf("Sum=%d\n",sum);
  return 0;
}
```

The output of the given program is

Output▼

7. Consider the following C function; **[GATE 2021]**

```
int SimpleFunction(int y[], int n, int x){
 int total=y[0], loopIndex;
 for(loopIndex=1;loopIndex<=n-1;loopIndex++)
    total=x*total+y[loopIndex];
 return total;
}
```

Output▼

Let $\mathbf{z}$ be an array of 10 elements with $\mathbf{z[i]} = \mathbf{1}$ for all $\mathbf{i}$ such that $\mathbf{0} \leq \mathbf{i} \leq \mathbf{9}$. State the value returned by the call **`SimpleFunction(z,10,2);`**

8. Write a function that takes two type int array input arguments and their effective size and produces a result array containing the sums of corresponding elements. For example, for the three-element input arrays 5 -1 7 and 2 4 -2 , the result would be an array containing 7 3 5 .

Space for Program ▼ **Output ▼**

9. The **bubble sort** is another technique for sorting an array. A bubble sort compares adjacent array elements and exchanges their values if they are out of order. In this way, the smaller values ”bubble“ to the top of the array (toward element 0), while the larger values sink to the bottom of the array. After the first pass of a bubble sort, the last array element is in the correct position; after the second pass the last two elements are correct, and so on. Thus, after each pass, the unsorted portion of the array contains one less element. Write and test a function that implements this sorting method.

Space for Program ▼ **Output ▼**

10. You have two independent sorted arrays of size m, and n respectively, where $m, n > 0$. You are required to merge the two arrays such that the merged array will be in sorted form and will contain exactly $m + n$ number of elements. You are not allowed to use any kind of sorting algorithm. Design your program to meet the above given requirement.

Example 1 :

First array:

12	20	24	32

Second array:

7	8	65	105

The merged sorted array:

7	8	12	20	24	32	65	105

Example 2 :

First array:

12	20	24

Second array:

7	8	65	105

The merged sorted array:

7	8	12	20	24	65	105

Example 3 :

First array :

12	20	24	100	120	130

Second array :

17	28	105	110

The merged sorted array:

12	17	20	24	100	105	110	120	130

NOTE :

Assume the elements of the array are non-negative integers. The elements can be read from the keyboard or can be generated randomly.

Space for Program **Output ▼**

Space for Program **Output ▼**

11. The *binary search* algorithm that follows may be used to search an array when the elements are in order. The algorithm for binary search given as;

    1. Let **bottom** be the subscript of the initial array element.
    2. Let **top** be the subscript of the last array element.
    3. Let **found** be false.
    4. Repeat as long as **bottom** isn't greater than **top** and the target has not been found
        5. Let **middle** be the subscript of the element halfway between **bottom** and **top**.
        6. if the element at **middle** is the target
                7. Set **found** to true and **index** to **middle**.

        else if the element at **middle** is larger than the target
                8. Let **top** be **middle - 1**.

        else
                9. Let **bottom** be **middle + 1**.

    Write and test a function **binary_srch** that implements this algorithm for an array of integers. When there is a large number of array elements, which function do you think is faster: **binary_srch** or the linear search algorithm.

Space for Program **Output ▼**

Space for Program **Output ▼**

12. Implement the following algorithm for linear search that sets a flag (for loop control) when the element being tested matches the target.

    1. Assume the target has not been found.
    2. Start with the initial array element.
    3. repeat while the target is not found and there are more array elements
        4. if the current element matches the target
                5. Set a flag to indicate that the target has been found.
        else
                6. Advance to the next array element.
    7. if the target was found
                8. Return the target index as the search result.
        else
                9. Return -1 as the search result.

    Create a user-defined function with prototype **`int linear_search(const int arr[ ], int target, int n);`** in your program to search the target element.

Space for Program **Output ▼**

Space for Program **Output ▼**

13. Write a program to copy the distinct elements of an int type array to another int type array. For example, if the input array is 4 7 7 3 2 5 5 then the output array will be 4 7 3 2 5.

Space for Program **Output ▼**

14. A barcode scanner for Universal Product Codes (UPCs) verifies the 12-digit code scanned by comparing the code's last digit (called a check digit) to its own computation of the check digit from the first 11 digits as follows:

    (1) Calculate the sum of the digits in the odd-numbered positions (the first, third, ..., eleventh digits) and multiply this sum by 3.

    (2) Calculate the sum of the digits in the even-numbered positions (the second, fourth, ..., tenth digits) and add this to the previous result.

    (3) If the last digit of the result from step 2 is 0, then 0 is the check digit. Otherwise, subtract the last digit from 10 to calculate the check digit.

    (4) If the check digit matches the final digit of the 12-digit UPC, the UPC is assumed correct.

    Write a program that prompts the user to enter the 12 digits of a barcode separated by spaces. The program should store the digits in an integer array, calculate the check digit, and compare it to the final barcode digit. If the digits match, output the barcode with the message "validated". If not, output the barcode with the message "error in barcode". Also, output with labels the results from steps 1 and 2 of the check-digit calculations. Note that the "first" digit of the barcode will be stored in element 0 of the array. Try your program on the following barcodes, three of which are valid. For the first barcode, the result from step 2 is 79 (0 + 9 + 0 + 8 + 4 + 0) * 3 + (7 + 4 + 0 + 0 + 5).

```
0 7 9 4 0 0 8 0 4 5 0 1
0 2 4 0 0 0 1 6 2 8 6 0
0 1 1 1 1 0 8 5 6 8 0 7
0 5 1 0 0 0 1 3 8 1 0 1
```

Space for Program **Output ▼**

Space for Program **Output ▼**

15. Write a program to grade an n-question multiple-choice exam (for n between 5 and 50) and provide feedback about the most frequently missed questions. Your program will take data from the file **`examdat.txt`**. The first line of the file contains the number of questions on the exam followed by a space and then an n -character string of the correct answers. Write a function fgetAnswers that inputs the answers from an open input file. Each of the lines that follow contain an integer student ID followed by a space and then that student's answers. Function fgetAnswers can also be called to input a student's answers. Your program is to produce an output file, **`report.txt`**, containing the answer key, each student's ID and each student's score as a percentage, and then information about how many students missed each question. Here are short sample input and output files.

examdat.txt

```
5 dbbac
111 dabac
102 dcbdc
251 dbbac
```

report.txt

```
                Exam Report
Question      1   2   3   4  5
Answer        d   b   b   a  c

ID        Score(%)
111        80
102        60
251        100

Question      1   2   3   4  5
Missed by     0   2   0   1  0
```

Space for Program **Output ▼**

Space for Program **Output ▼**