LABORATORY MANUAL
B.SC. III SEMESTER
(ELECTRONICS)

Department of Pure & Applied Physics

Guru Ghasidas Vishwavidyalaya

(A Central University)

C/C++ Programming and Data Structures Lab

Name of the experiments:

1. Find minimum and maximum of N numbers.

2. Calculate the value of sin (x) and cos (x) using the series. Also print sin (x) and cos (x) value using library function.

3. Generate and print prime numbers up to an integer N.

4. Calculate the subject wise and student wise totals and store them as a part of the structure.

5. Create a stack and perform Pop, Push. Traverse operations on the stack using Linear Linked list.

6. Create circular linked list having information about a college and perform Insertion at front, Deletion at end.

7. Create a Linear Queue using Linked List and implement different operations such as Insert, Delete, and Display the queue elements.

8. Implement Insertion sort and Merge sort.

9. Implementation of Bubble sort and Selection sort.

Experiment 1

Objective: Find the minimum and maximum of N numbers.

Theory: The minimum and maximum value from given set of N numbers can be found using control statement. A C++ control statement redirects the flow of a program in order to execute additional code. These statements come in the form of conditionals (if-else, switch) and loops (for, while, do-while). Each of them relies on a logical condition that evaluates to a Boolean value in order to run one piece of code over another.

If statement executes the code in the body section if the condition evaluates to True, otherwise it skips the body. When the condition evaluates to False, the program will either test another condition with a following Else-if, run the code inside an Else statement, or continue as normal if neither an Else-if nor Else block exist.

A for loop allows for a block of code to be executed until a conditional becomes false. For loops are usually used when a block of code needs to executed a fixed number of times. Each loop consists of 3 parts, an initialization step, the conditional, and an iteration step. The initialization is run before entering the loop, the condition is checked at the beginning of each run through the loop (including the first run), and the iteration step executes at the end of each pass through the loop, but before the condition is rechecked. It is usual practice to have the iteration step move the loop on step closer to making the condition false, thus ending the loop, but this does not need to be the case.

Algorithm: the algorithm is as follows:

1. Initialize the integer/float type array of 10 numbers.
2. Define the two variables max & min and initialize them to zero each.
3. Create a 'for loop' to find out each number in the array and find out whether the number is more or less than max and min.
4. Create a 'If statement' for max and min numbers. (If above statement is true, change the value of max with the number. Do same for min)
5. At the end of the program display the content of max (for maximum value) and min (for minimum value) as an output.

Precautions:

1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.

Experiment 2

Objective: Calculate the value of sin (x) and cos (x) using the series also print sin (x) and cos (x) value using library function.

Theory: Sine and cosine (sin and cos) are trigonometric functions that are used in many real-life applications. their expansion are as follows:

Sin(x)= $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \ldots\ldots\ldots..$

Cos(x)= $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \ldots\ldots\ldots..$

for writing C++ program basic operators are used like float; pow etc.

float: The float data type is used to store floating-point numbers. Floating-point numbers are numbers that have a decimal point, such as 3.14 or 10.5.

Pow: It is used to perform the power operation, which is raising one number (the base) to the power of another (the exponent). This function takes two arguments and returns the result as a double .

There are some pre-written sets of code that performs a specific task and is included in the C++ Standard Library known as library function. These functions are designed to be reusable and cover a wide range of common operations. Library functions simplify programming tasks by providing ready-made solutions for frequently encountered problems. To use a library function in C++, typically include the necessary header file with the #include statement. Here, for this program the library function for sin(x) is 1 and cos (x) is 0.

Algorithm: the algorithm are as follows:

1. Use #Include<cmath> to initialize program.
2. Accept the value of x from user.
3. Use the series formula for sin(x) mentioned above by applying pow(), for example for writing x^2 write pow(x,2).
4. Similarly, use cos(x) series formula for calculating value.
5. Use sin(x) & cos(x) library functions.
6. Compare the calculated value with pre-defined value.

Precaution:

1. Write program carefully.
2. The output value should be approximately equal to library function.
3. Apply proper syntax.

Experiment 3

Objective: Generate and print prime numbers up-to an integer N.

Theory: A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. Prime number from given set of numbers can be find by using C++ control statements. A C++ control statement redirects the flow of a program in order to execute additional code. These statements come in the form of conditionals (if-else, switch) and loops (for, while, do-while). Each of them relies on a logical condition that evaluates to a Boolean value in order to run one piece of code over another.

If statement executes the code in the body section if the condition evaluates to True, otherwise it skips the body. When the condition evaluates to False, the program will either test another condition with a following Else-if, run the code inside an Else statement, or continue as normal if neither an Else-if nor Else block exist.

A for loop allows for a block of code to be executed until a conditional becomes false. For loops are usually used when a block of code needs to executed a fixed number of times. Each loop consists of 3 parts, an initialization step, the conditional, and an iteration step. The initialization is run before entering the loop, the condition is checked at the beginning of each run through the loop (including the first run), and the iteration step executes at the end of each pass through the loop, but before the condition is rechecked. It is usual practice to have the iteration step move the loop on step closer to making the condition false, thus ending the loop, but this does not need to be the case. Nested loop condition in C++ is used for iterating over multiple sequences of data. It is a powerful tool that can be used to solve a variety of problems.

Algorithm:

1. Initialize the integer/float assign the value as zero i.e., ctr=0
2. Input the limits of number till which the series of prime number is to printed.
3. Create a nested for loop to compile up to N numbers.
4. Give condition to print prime numbers inside the nested loop.
5. Create a control statement if for continuity of program (remainder=0, increase ctr value by 1).
6. Prime number will be detected foe ctr=2.
7. Output will be obtained as all prime number from the series of N numbers.

Precaution:

1. Use syntax carefully.

Experiment 4

Objective: Calculate the subject wise and student wise totals and store them as a part of the structure.

Theory: A structure is a user-defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type. The struct keyword is used to define the structure in the C programming language. The items in the structure are called its member and they can be of any valid data type. Structure is used to represent student information, including student ID, student name, and an array to store subject-wise marks. All variables use data type during declaration to restrict the type of data to be stored. Therefore, data types are used to tell the variables the type of data they can store. Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data type with which it is declared. Every data type requires a different amount of memory. The primary or built-in or fundamental data type in C++ includes: Integer, character, Boolean, Floating Point, Double Floating Point, Valueless or Void, Wide Character.

Algorithm:

1. Define a structure to represent student information, including:

   - Student ID
   - Student name
   - Array to store subject-wise marks
2. Read the number of students (S) and the number of subjects (N) from the user.
3. Declare an array of structures to store information for each student.
4. Initialize a loop for each student (s) from 0 to S-1:
   a) Read and store the student ID and name in the structure.
   b) Initialize a loop for each subject (n) from 0 to N-1: Read and store the marks for each subject in the structure.
5. Initialize an array to store subject-wise totals for each subject.
6. Initialize a loop for each subject (n) from 0 to N-1:
   a) Initialize the subject-wise total to 0.
   b) Iterate through each student (s) and add the marks for the current subject to the total.
   c) Store the subject-wise total in the array.
7. Initialize a loop for each student (s) from 0 to S-1:
   a) Initialize the student-wise total to 0.
   b) Iterate through each subject (n) and add the marks for the current subject to the total.
   c) Store the student-wise total in the structure.
8. Print the subject-wise totals for each subject.
9. Print the student-wise totals along with student information.
10. End of the algorithm.

Precautions:

1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.

Experiment 5

Objective: Create as stack and perform a Pop, Push, Traverse operations on the stack using linear linked list.

Theory: Push operation on stack implementation using linked-list involves several steps: Create a node first and allocate memory to it. If the list is empty, then the node is pushed as the first node of the linked list. This operation assigns a value to the data part of the node and gives NULL to the address part of the node. The popping node from the linked list is different from the popping element from the array. To perform the pop operation. the node is removed from the end of the linked list. Therefore, must delete the value stored in the head pointer, and the node must get free. The following link node will become the head node now. An underflow condition will occur when we try to pop an operation when the Stack is already empty. The Stack will be meaningless if the head pointer of the list points to NULL. Traversal operation in linked list means accessing all the elements of the list starting from the head node to the last node. Traversing a tree and graph means visiting every node at least once.

Algorithm:

Push()
i) Create a new Node with the given data.
ii) Check whether the stack is empty (TOP==NULL).
iii) If it is **e**mpty, then set he pointer of the node toNULL.
iv) If it is not empty, then make the node point to TOP.
v) Finally, make the new Node as TOP.

Algorithm_push()

```
if (TOP == NULL)
newNode -> next = NULL
else
newNode -> next = TOP
```

pop()
i) Check whether stack is empty **(**top == NULL**).**
ii) If it is empty, then display "EMPTY STACK"
iii) If it is not empty, then create a temporary node and set it to TOP.
    a. Print the data of TOP.
    b. Make TOP to point to the next node.
    c. Delete the temporary node.

Algorithm_pop()

```
if TOP == NULL
print "EMPTY STACK"else
create a temporary node, temp = top
```

```
print TOP -> data
TOP = TOP -> next
free(temp)
```

Traverse()

1. Create an empty stack (say S).
2. Initialize the current node as root.
3. Push the current node to S and set current = current->left until current is NULL
4. If current is NULL and the stack is not empty then:
    - Pop the top item from the stack.
    - Print the popped item and set current = popped_item->right
    - Go to step 3.
5. If current is NULL and the stack is empty then we are done.

Precaution:

1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.

Experiment 6

Objective: Create circular linked list having information about a college and perform Insertion at front, Deletion at end.

Theory: The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end. We can do Insertion & Deletion operations on the circular linked list. The first node of the linked list is called the head node. It is the starting point of a linked list.

Algorithm:

i) Initialize Head

- Create a variable to represent the head of the circular linked list.

ii) Insert at Front

- Decision: Check if the list is empty.

  If yes, create a new node and set it as the head

  If no, create a new node and link it to the existing list, then update the head

iii) Display List

- Display the contents of the circular linked

iv) Deletion at End

- Decision: Check if the list is empty.

  If yes, display an error message.

  If no, traverse the list to find the last node and delete it.

v) Display Updated List

- Display the contents of the circular linked list after deletion.

vi) End

Precaution:

1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.

Experiment 7

Objective: Create a linear queue using linked list and implement different operations such as insert, Delete and display the queue element.

Theory: the array implementation cannot be used for the large-scale applications where the queues are implemented. One of the alternatives of array implementation is linked list implementation of queue. The storage requirement of linked representation of a queue with n elements is o(n) while the time requirement for operations is o(1). In a linked queue, each node of the queue consists of two parts i.e. data part and the link part. Each element of the queue points to its immediate next element in the memory. In the linked queue, there are two pointers maintained in the memory i.e. front pointer and rear pointer. The front pointer contains the address of the starting element of the queue while the rear pointer contains the address of the last element of the queue. Insertion and deletions are performed at rear and front end respectively. If front and rear both are NULL, it indicates that the queue is empty. The insert operation appends the queue by adding an element to the end of the queue. The new element will be the last element of the queue. Deletion operation removes the element that is first inserted among all the queue elements.

Algorithm:

i) Define a structure to represent a node in the linked list:
- Data: to store the queue element
- Next: a pointer to the next node in the linked list

ii) Define a structure for the queue:
- Front: a pointer to the front (head) of the linked list
- Rear: a pointer to the rear (tail) of the linked list

iii) Initialize an empty queue by setting Front and Rear to NULL.

iv) Implement the insert operation (enqueue):
- Create a new node with the given data.
- If the queue is empty (Front and Rear are NULL), set Front and Rear to the new node.
- Otherwise, set the Next of the current Rear to the new node and update Rear to the new node.

v) Implement the delete operation (dequeue):
- If the queue is empty (Front is NULL), print an error message.
- Otherwise, store the data from the node at the Front, move Front to the next node, and free the memory of the removed node.

vi) Implement the display operation:
- Traverse the linked list from Front to Rear.
- Print each element in the queue.

vii) End of the algorithm.

Precaution:
1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.

Experiment 8

Objective: Implement Insertion sort and Merge sort.

Theory: Merge sort is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array. In simple terms, we can say that the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together. This process is repeated until the entire array is sorted. Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part. Merge Sort is preferred for huge data sets. It happens to compare all the elements present in the array hence is not much helpful for small datasets, whereas Insertion Sort is preferred for fewer elements. It becomes fast when data is already sorted or nearly sorted because it skips the sorted values. Considering average time complexity of both algorithm we can say that Merge Sort is efficient in terms of time and Insertion Sort is efficient in terms of space.

Algorithm:

Algorithm of Insertion sort-

i) Declare an array of size, n.
ii) Provide the n inputs such that the array (named arr) is unsorted.
iii) Run a loop, with i from 0 to n-1
iv) Assign arr[i] as key and i-1 as j
v) While j >= 0 and arr[j+1] > arr[j] is True
vi) arr[j+1] = arr[j]
vii) Increment j = j + 1
viii) Assign key at a[j+1]

Algorithm of Merge sort-

Step 1: **Start**
Step 2: Declare an array and left, right, mid variable
Step 3: Perform merge function.

```
mergesort(array,left,right)
mergesort (array, left, right)
if left > right
return
mid= (left+right)/2
mergesort(array, left, mid)
mergesort(array, mid+1, right)
merge(array, left, mid, right)
```

Step 4: Stop

Precaution:

1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.

Experiment 9

Objective: Implementation of bubble sort and selection sort.

Theory: Selection sort is one sorting technique used for sorting the array. In selection sort, an array is divided into two sub- arrays, i.e., one is an unsorted sub-array, and the other is sorted sub-array. Initially, we assume that the sorted subarray is empty. First, we will find the minimum element from the unsorted subarray, and we will swap the minimum element with an element which is at the beginning position of the array. This algorithm is named as selection sort because it is selecting the minimum element and then performs swapping. The bubble sort is also one of the sorting techniques used for sorting the elements of an array. The basic principle behind the bubble sort is that the two adjacent elements are to be compared; if those elements are in correct order, then we move to the next iteration. Otherwise, we swap those two elements.

Algorithm:

Algorithm for Bubble Sort

Let's assume that arr is an array with n members (elements):

i) begin BubbleSort(arr)
ii) for all array elements
iii) if arr[i] > arr[i+1]
iv) swap(arr[i], arr[i+1])
v) end if
vi) end for
vii) return arr
viii) end BubbleSort

Algorithm for Selection Sort

Declare an array of size, n.

i) Provide the n inputs such that the array is unsorted.
ii) Loop i, from 0 to n-2
iii) Inner loop j runs from i+1 to n-1

1. On each iteration of j:

- If arr[i] > arr[i] then we swap the number in order to sort the array in ascending order.
- Else continue

2. Print the sorted array

In this manner, the array will be sorted in ascending order will the lowest element at the start and the largest at the end position. Looking at the algorithm, its gives an idea to do the selection sort program in c.

Precaution:

1. Write programs carefully.
2. Apply proper control statements.
3. Use proper syntax.