**THEORY OF EACH DS EXPERIMENT LAB FILE**

**Exp 1(a):**

**Algorithm:**

1. In main (), input the array from the user and also the item to be searched.
2. Using a simple for loop with an if statement check whether the arr[i]==num, if yes, make flag=1 and pos = i
3. Lastly using an if-else condition check the value of flag (either 0 or 1) to diplay the final result.

**Learning Outcome:** I learnt to develop a code that demonstrates the working of linear search.

**Exp 1(b):**

**Algorithm:**

1. Declare a function bsearch() to perform the binary search. Make sure to include variables like beg, last and mid and use if, else-if and else statements in a while loop to perform the search.
2. In the function return the position (mid) at which the element is found, or return -1 if element is not found in the array entered by the user.
3. In main (), input the array from the user and also the item to be searched.
4. Once the function is called the returned value can be checked by simple if-else to display the final result.

**Learning Outcome:** I learnt to develop a code that demonstrates the working of binary search.

**Exp 2:**

1. Create a stack and add elements to it by taking user’s input for every element. Initialise the top as -1.
2. Input the choice from the user for the function to be performed on the stack using a switch-case method.

**Algorithm for Peek:**

1. If top=-1, return Underflow, if not, return the last inserted element (element at the top of the stack).

**Algorithm for Push:**

1. If top=number of elements user wanted in the stack, then print stack is full (Overflow), if not, ask the user for the value to be pushed into the stack and assign the top position to the element entered.

**Algorithm for Pop:**

1. If top=-1, return Underflow, if not, store the element at the top of the stack in another variable to print and reduce the value of top by 1.

**Algorithm for Display:**

1. If top=-1, return Underflow, if not, traverse the stack using a loop and print the values in the stack for each index of the loop.

**Learning Outcome:**

**Exp 3(a):**

**Algorithm for Bubble Sort:**

1. In main () create a loop to input the array elements and print the unsorted as well as sorted array after calling the bubble sort function.
2. In the BubbleSort() function, create a temp variable along with a loop to traverse the entire array.
3. On finding a value lower than the previous one in the array use the temp variable to swap the two and proceed further until the array is sorted.

**Algorithm for Insertion Sort:**

1. In main () create a loop to input the array elements and print the unsorted as well as sorted array after calling the bubble sort function.
2. In the InsertionSort() function, create a for loop to traverse the array and while loop to place the found smaller element at the correct position in the array.
3. Insert the smaller element to the appropriate position, swapping the element present there previously with the smaller element.

**Learning Outcome:** I learnt to implement Bubble and Insertion Sorting Techniques in C Language.

**Exp 3(b):**

**Algorithm for Selection Sort:**

1. In main () create a loop to input the array elements and print the unsorted as well as sorted array after calling the bubble sort function.
2. In the SelectionSort() function, create a for loop to traverse the array and compare the first element of the array with every element.
3. If a smaller element is found compare that with the further elements in order to find the smallest element.
4. Place the smallest element at the first position (0 index) after first traversal and so on to achieved the sorted array.

**Algorithm for Merge Sort:**

1. In main () create a loop to input the array elements and print the unsorted as well as sorted array after calling the bubble sort function.
2. Create Merge() & Merge Sort() functions, making sure to call MergeSort() and Merge() in the function definition for the MergeSort() function.
3. This creates a recursive function in order to divide the array into smaller parts and then sort them using backtracking and the classic divide and conquer approach.
4. Lastly, merge all the sorted array parts in order to achieve the final array in sorted format.

**Learning Outcome:** I learnt to implement Selection and Merge Sorting Techniques in C Language.

**Exp 4:**

1. Create the linked list as a global declaration/variable using the struct keyword and name it by a pointer named top.
2. Input the choice of the user for the function to be performed on the linked list.

**Algorithm for push:**

1. Using malloc assign a dynamic memory to a temporary pointer, and to this memory store the data in the list and at its link add NULL.
2. Check if there is another element in the list by top=NULL, if yes, make tmp=top, if no, make tmp->link=top and top=tmp.

**Algorithm for display:**

1. Check if the top is NULL, if yes, return Underflow, else, traverse the list using a while loop with the condition tmp!=NULL, making tmp=tmp->link, to print the value for each iteration

**Algorithm for pop:**

1. Check if the top is NULL, if yes, return Underflow, else, pop the element at the top and make the top one element lower in the list.

**Exp 5:**

1. Create the linked list as a global declaration/variable using the struct keyword and name it by pointers f and r.
2. Input the choice of the user for the function to be performed on the linked list.

**Algorithm for enqueue:**

1. Using malloc assign a dynamic memory to a temporary pointer, and to this memory store the data in the list and at its link add NULL.
2. Check if there is another element in the list by f and r=NULL, if yes, make f and r=tmp, if no, make r->link=tmp and r=tmp.

**Algorithm for display:**

1. Check if f=NULL, if yes, return Underflow, else, traverse the list using a while loop with the condition tmp!=NULL, making tmp=tmp->link, to print the value for each iteration

**Algorithm for dequeue:**

1. Check if f=NULL, if yes, return Underflow, else, remove the element at the front and make the front one element higher in the list using f=f->link.

**Exp 6:**

**Algorithm:**

1. Start by creating a Linked List of the name student and add the details required by making a simple create function to create the initial node.
2. For insertion at position, traverse the list to find the correct position, if the position is NULL, its invalid, else add the new record from tmp to that position.
3. Create a display function to display the Linked List.

**Exp 7:**

**Algorithm:**

1. Create a Doubly Linked List with name Employee and having details of name, employee ID, department, designation and salary.
2. Create a switch case for inputting the user’s choice to perform either creation/insertion at the beginning/ deletion at the end/ display or exit the program.
3. For creation use malloc to allocate memory to the temporary variable and input the required employee details. Add the node to the employee linked list by making start=tmp.
4. For Insertion at the beginning input all the values and make the tmp node as the first node of the DLL.
5. For deletion at the end of the DLL, first check if the list is empty, if yes, print no element in linked list, else traverse the linked list and delete the last element by using free(tmp).
6. To display the Linked List, first check if its empty, if not, then traverse and print each detail of employee using while loop.
7. If any invalid choice is entered, print Invalid Choice in the switch case and exit the program.

**Exp 8:**

**Algorithm:**

1. Create a Circular Linked List with name College and having details of College Name, College ID and Number of students in the college.
2. Create a switch case for inputting the user’s choice to perform either creation/insertion at the beginning/ deletion at the end/ display or exit the program.
3. For creation use malloc to allocate memory to the temporary variable and input the required employee details. Add the node to the employee linked list by making start=tmp and tmp->next=start.
4. For Insertion at the beginning input all the values and make the tmp node as the first node of the CLL.
5. For deletion at the end of the CLL, first check if the list is empty, if yes, print no element in linked list, else traverse the linked list and delete the last element by using free(tmp).
6. To display the Linked List, first check if its empty, if not, then traverse and print each detail of employee using while loop.
7. If any invalid choice is entered, print Invalid Choice in the switch case and exit the program.