Course Hand Out

Subject Name: Data Structures using C

Prepared by: Mrs. J. Priya, Assistant Professor, CSE

Year, Semester, Regulation: II Year- I Sem (R18)

UNIT –I

KEY POINTS:

ADT:

Abstract Data type (ADT) is a type (or class) for objects whose behaviour is defined by a set of value and a set of operations.The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called “abstract” because it gives an implementation-independent view. The process of providing only the essentials and hiding the details is known as abstraction.

Linked list:

A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head is NULL.Each node in a list consists of at least two parts:
1) Data
2) Pointer (Or Reference) to the next node

Delete an Element from single linked list

```
void delete1()

{

        struct node *p=root,*temp=root,*q;

        int loc,len,i=1;

        printf("\nENTER THE LOCATION TO DELETE\n");

        scanf("%d", &loc);

        len=length();

        if(loc>len)

        printf("THE LIST CONTAINS ONLY %d NODES\n",len);

        else if(loc==1)
```

```
                {
                        root=temp->link;
                        temp->link=NULL;
                        free(temp);
                }
                else
                {
                        while(i<loc-1)
                        {
                                p=p->link;
                                i++;
                        }
                        q=p->link;
                        p->link=q->link;
                        q->link=NULL;
                        free(q);
                }
                display();
        }
```

Display for Single linked list

```
void display()
{
        struct node * temp;
        temp=root;
        if(temp==NULL)
        printf("\nLINKED LIST IS EMPTY\n");
        else
        {
                while(temp!=NULL)
```

```
        {
                printf("%d->",temp->data);
                temp=temp->link;
        }
   }
}
```

Insertion at Particular location in Single Linked list

```
void addatafter()
{
        struct node * temp,*p;
        int len,i=2,loc;
        printf("\nENTER A LOCATION TO INSERT\n");
        scanf("%d",&loc);
        len=length();
        if(loc>len)
        {
                printf("LINKED LIST CONTAINS ONLY %d NODES\n", len);
        }
        else
        {
                temp=(struct node*)malloc(sizeof(struct node));
                printf("\nENTER NEW NODE'S DATA\n");
                scanf("%d",&temp->data);
                temp->link=NULL;
                p=root;
                while(i<loc)
                {
                        p=p->link;
```

```
                    i++;
                }
                temp->link=p->link;
                p->link=temp;
            }
            display();
}
```

Difference between Single Linked List and Double Linked List

SINGLY LINKED LIST (SLL)	DOUBLY LINKED LIST (DLL)
SLL has nodes with only a data field and next link field.	DLL has nodes with a data field, a previous link field and a next link field.
In SLL, the traversal can be done using the next node link only.	In DLL, the traversal can be done using the previous node link or the next node link.
The SLL occupies less memory than DLL as it has only 2 fields.	The DLL occupies more memory than SLL as it has 3 fields.
Less efficient access to elements.	More efficient access to elements.

Stack ADT:

A stack is an Abstract Data Type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack, for example – a deck of cards or a pile of plates, etc. A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, we can only access the top element of a stack. This feature makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last, is accessed first. In stack terminology, insertion operation is called **PUSH** operation and removal operation is called **POP** operation.

Push Operation in Stack:

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps −

- **Step 1** − Checks if the stack is full.
- **Step 2** − If the stack is full, produces an error and exit.
- **Step 3** − If the stack is not full, increments **top** to point next empty space.
- **Step 4** − Adds data element to the stack location, where top is pointing.
- **Step 5** − Returns success.

Pop Operation in Stack

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead **top** is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element and deallocates memory space.

A Pop operation may involve the following steps −

- **Step 1** − Checks if the stack is empty.
- **Step 2** − If the stack is empty, produces an error and exit.
- **Step 3** − If the stack is not empty, accesses the data element at which **top** is pointing.
- **Step 4** − Decreases the value of top by 1.
- **Step 5** − Returns success.

Applications of stack:

- Balancing of symbols
- Infix to Postfix /Prefix conversion
- Redo-undo features at many places like editors, photoshop.
- Forward and backward feature in web browsers
- Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, histogram problem.
- Other applications can be Backtracking, Knight tour problem, rat in a maze, N queen problem and sudoku solver
- In Graph Algorithms like Topological Sorting and Strongly Connected Components

Algorithm for conversion of infix to postfix expression

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, output it.
3. Else,
…..**3.1** If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty or the stack contains a '('), push it.
…..**3.2** Else, Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. After doing that Push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
4. If the scanned character is an '(', push it to the stack.
5. If the scanned character is an ')', pop the stack and and output it until a '(' is encountered, and discard both the parenthesis.
6. Repeat steps 2-6 until infix expression is scanned.
7. Print the output
8. Pop and output from the stack until it is not empty.

QUEUE ADT

The queue abstract data type (ADT) follows the basic design of the stack abstract data type. A Queue contains elements of the same type arranged in sequential order. Operations take place at both ends, insertion is done at the end and deletion is done at the front. Following operations can be performed:

- enqueue() – Insert an element at the end of the queue.
- dequeue() – Remove and return the first element of the queue, if the queue is not empty.
- peek() – Return the element of the queue without removing it, if the queue is not empty.

- size() – Return the number of elements in the queue.
- isEmpty() – Return true if the queue is empty, otherwise return false.
- isFull() – Return true if the queue is full, otherwise return false.

University Exam Questions:

Short Answers:

1. What is ADT? (Dec-17, Marks 2)
2. Define Node in Single linked list. (Dec-17,Marks 2)
3. Define Stack (Dec-18,16, May-06 , Marks 2)
4. Define Queue (May-17,Dec-06,16 Marks 2)
5. Define Double Linked List (May-09,15,Dec-12,16, Marks 2)
6. What is the Difference between singly linked list and doubly linked list (May-09, Marks 3)
7. What are all the operations of Circular Queue (Nov-15,May-15, Marks 3)
8. Explain the functions of Stack (Jan-14,Marks 3)
9. What is stack? List the applications of stack (Dec-16,Marks 3)
10. What are all the steps in evaluating the postfix expression? (Dec-16, Marks 3)
11. Evaluate the Postfix expression 123+*321-+* step by step.
12. Write any four applications of Queue. (May-08,16, Marks 2)

Long Answers:

1. Write a program to delete an element from single linked list. (dec-17, Marks 5)
2. Given a Singly linked list, write a function to swap every two nodes for example 1->2->2->3->4->5->6 Should become 2->1->4->3->6->5.(Dec-14,Marks 10)
3. Write a C program to implement insertion to the immediate left of k^{th} node in singly linked list. (Dec-16,Marks 5)
4. What is the Difference between singly linked list and doubly linked list (May-16, Marks 5)
5. Write a functions and explain for Push and Pop operations of a Stack (Jan-14, Marks 7)
6. Write a program to push the element to the push operation into a stack, (Dec-17, Marks 5)
7. Write an algorithm to convert infix to postfix expression. (Dec-17, May-15, Dec-17,18, Marks 5)
8. Explain the operations of Queue with an example.(Nov-15,Marks 5)
9. Discuss about linked list implementation of queue ADT. (Dec-18, Marks 5)
10. Write About Stack ADT (May-09, Marks 5)

Fill in the Blanks:

1. Primitive data structures are generally _____ data types in programming language.
2. A list with no nodes is called _____
3. Linked list uses NULL pointer to signal _____
4. Each node in the double linked list will have _____ fields.
5. Stack can be implemented by using either _____ or _____
6. The removing of an element from an stack is called _____
7. The insertion of an element on to stack is called _____
8. The postfix expression for a*b+c is _____
9. Polish notation is often called as _____
10. ADT stands for _____
11. After every insertion in stack top will be _____ by one.
12. There are _____ ends in Queue.
13. Reverse polish notation is also called as _____
14. Tree is an example of _____ type of data structure.
15. The data structure used for evaluating a postfix expression is _____