Techno India NJR Institute of Technology

Course File

Computer Programming Lab-I

(3EC3-24)

Mr. Pankaj Ameta
(Assistant Professor)
Department of ECE

3EC3-24: Computer Programming Lab-I

1 Credit **Max. Marks: 50 (IA:30, ETE:20)**
0L:0T:2P

1.	Write a simple C program on a 32 bit compiler to understand the concept of array storage, size of a word. The program shall be written illustrating the concept of row major and column major storage. Find the address of element and verify it with the theoretical value. Program may be written for arrays upto 4-dimensions.
2.	Simulate a stack, queue, circular queue and dequeue using a one dimensional array as storage element. The program should implement the basic addition, deletion and traversal operations.
3.	Represent a 2-variable polynomial using array. Use this representation to implement addition of polynomials.
4.	Represent a sparse matrix using array. Implement addition and transposition operations using the representation.
5.	Implement singly, doubly and circularly connected linked lists illustrating operations like addition at different locations, deletion from specified locations and traversal.
6.	Repeat exercises 2, 3 & 4 with linked structures.
7.	Implementation of binary tree with operations like addition, deletion, traversal.
8.	Depth first and breadth first traversal of graphs represented using adjacency matrix and list.
9.	Implementation of binary search in arrays and on linked Binary Search Tree.
10.	Implementation of insertion, quick, heap, topological and bubble sorting algorithms.

Course Outcomes:

Course Code	Course Name	Course Outcomes	Details
3EC3-24	**Computer Programming Lab-I**	**CO1**	Understand the linear and non-linear data structure.
		CO2	Implement the concept of data structures such as stacks, queues and linked lists.
		CO3	Apply the concept of arrays and linked list to implement operations on sparse matrix and polynomials.
		CO4	Implement the nonlinear data structure such as tree and graph.
		CO5	Analyze various kinds of searching and sorting techniques.

Course Outcome Mapping with Program Outcome:

Course Outcome	**Program Outcomes (PO's)**											
CO. NO.	**Domain Specific**					**Domain Independent**						
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	2	1	1	1	3				1	1		1
CO2	1	2	2	2	2				2	2		2
CO3	2	2	1	2	1				2	2		3
CO4	1	2	2	2	1				3	2		2
CO5	2	2	2	2	2				2	3		2
1: Slight (Low) , 2: Moderate (Medium), 3: Substantial (High)												

List of Practical's

1. Write a program to reverse a number (user input).
2. Write a program to reverse a number using array.
3. Write a program to perform bubble sort.
4. Write a program to perform selection sort.
5. Write a program to perform insertion sort.
6. Write a program to perform insertion, deletion, merge and display the array.
7. Write a program to perform addition, subtraction, multiplication of a matrix.
8. Write a program to perform implementation of sparse matrix.
9. Write a program to perform insertion, insertion at position, deletion, deletion from a position and displaying a linked list.
10. Write a program to perform implementation of stack.
11. Write a program to perform implementation of stack using linked list.
12. Write a program to perform implementation of queue.
13. Write a program to perform implementation of queue using linked list.
14. Write a program to perform linear and binary search.
15. Write a program to implement doubly linked list.
16. Write a program to perform implementation of polynomial expression and perform addition of two polynomial expressions.

1. **Write a program to reverse a number (user input).**

   Code:

```
#include<stdio.h>
#include<conio.h>
int revint(int);
void main()
{
        int n,result;
        printf("Eneter the no");
        scanf("%d",&n);
        result=revint(n);
        printf("The reverse of number %d is %d",n,result);
        getch();
}
int revint(int p)
{
        int d;
        static int r=0;
        if(p==0)
        {
                return(p);
        }
        else
        {
                d=p%10;
                r=r*10+d;
                revint(p/10);
        }
        return(r);
}
```

2. Write a program to reverse a number using array

Code :

```
#include <stdio.h>
int main()
{
        int array[100], n, c, t, end;
        scanf("%d", &n);
         end = n – 1;
        for (c = 0; c < n; c++) {
        scanf("%d", &array[c]);
        for (c = 0; c < n/2; c++)
        {
                t = array[c];
                array[c] = array[end];
                array[end] = t;
                end--;
        }
        printf("Reversed array elements are:\n");
        for (c = 0; c < n; c++)
        {
                 Printf("%d\n", array[c]);
        }
        return 0;
}
```

3. Write a program to perform bubble sort.

Code:

```
#include<stdio.h>
void main()
{
        int a[100],n,i,j,temp;
        printf("How many elements");
        scanf("%d",&n);
        printf("Enter the element of array");
        for(i=0;i<=n-1;i++)
        {
                scanf("%d",&a[i]);
        }
        for(i=0;i<=n-1;i++)
        {
                for(j=0;j<=n-1-i;j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
        printf("Element of array after the sorting are:\n");
        for(i=0;i<=n-1;i++)
        {
                printf("%d\n",a[i]);
        }
}
```

4. Write a program to perform selection sort.

Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[100],n,i,j,temp,loc,min;
        clrscr();
        printf("\How many elements:\n");
        scanf("%d",&n);
        printf("Enter the element of array\n");
        for(i=0;i<=n-1;i++)
        {
                scanf("%d",&a[i]);
        }
        min=a[0];
        for(i=0;i<=n-1;i++)
        {
                min=a[i];
                loc=i;
                for(j=i+1;j<=n-1;j++)
                {
                        if(a[j]<min)
                        {
                                min=a[j];
                                loc=j;
                        }
                }
```

```
            if(loc!=1)
            {
                temp=a[i];
                a[i]=a[loc];
                a[loc]=temp;
            }
        }

        printf("The number after selection sorting are:\n");
        for(i=0;i<=n-1;i++)
        {
            printf("%d\n",a[i]);
        }
        getch();
}
```

5. Write a program to perform insertion sort

Code :

```
#include<stdio.h>
void main()
{
        int a[100],n,k,i,j,temp;
        printf("How many elements\n");
        scanf("%d",&n);
        printf("Enter the element of array");
        for(i=0;i<=n-1;i++)
        {
                scanf("%d",&a[i]);
        }
        for(k=1;k<=n-1;k++)
        {
                temp=a[k];
                j=k-1;
                while((temp<a[j])&&(j>=0))
                {
                        a[j+1]=a[j];
                        j=j-1;
                }
                a[j+1]=temp;
        }
        printf("Element of array after sorting\n");
        for(i=0;i<=n-1;i++)
        {
                printf("%d\n",a[i]);
        }
}
```

6. Write a program to perform insertion, deletion , merge and display the array.

Code:

```
#include<stdio.h>
#include<conio.h>
void insert();
void del();
void traversa();
int a[10],i=-1;
void main()
{
        int n;
        clrscr();
        do
        {
                printf("****************************** YOUR CHOICES
                ARE ***************************\n");
                printf("1. Insert\n");
                printf("2. Delete\n");
                printf("3. Traverse\n");
                printf("4. Exit\n");
                printf("NOW ENTER YOUR CHOICE :\n");
                scanf("%d",&n);
                switch(n)
                {
                        case 1:
                                insert();
                                break;
                        case 2:
                                del();
                                break;
```

```
                        case 3:
                                traversa();
                                break;
                        case 4:
                                break;
                        default:
                                printf("SORRY!YOUR CHOICE IS INVALID!!");
                }
        }while(n!=5);
        getch();
}
void insert()   /Body of insert function
{
        if(i<=8)
        {
                printf("Enter the No.(10 elements)\n");
                while(i<=8)
                {
                        i++;
                        scanf("%d",&a[i]);
                }
        }
        else
        {
                printf("The Array is Full");
        }
}
void del()  //Body of del function
{
        int item,j,flag=0,pos;
        printf("Enter the no to delete : ");
```

```
        scanf("%d",&item);
        for(j=0;j<=i;j++)
        {
                if(a[j]==item)
                {
                        flag=1;
                        pos=j;
                }
        }
        if(flag)
        {
                for(j=pos;j<=i;j++)
                {
                        a[j]=a[j+1];
                }
                i--;
        }
        else
        {
                printf("The no is not in list, Sorry\n");
        }
}
void traversa()  //Body of traversa function
{
        int j;
        for(j=0;j<=i;j++)
        {
                printf("%d\n",a[j]);
        }
}
```

7. Write a program to perform addition ,subtraction,multiplication of a matrix

Code:

```
#include<stdio.h>
void main( )
{
        int a[2][2], b[2][2],s[2][2];
        int i,j,k;
        printf("Enter first matrix:\n" );
        for( i=1;i<=2;i++)
        {
                for( j=1;j<=2;j++)
                {
                        printf("Enter%d%d\n",i,j);
                        scanf("%d",&a[i][j]);
                }
        }
        printf("Enter second matrix:\n");
        for(i=1;i<=2;i++)
        {
                for(j=1;j<=2;j++)
                {
                        printf("Enter %d%d\n",i,j);
                        scanf("%d",&b[i][j]);
                }
        }
        for (i=1;i<=2;i++)
        {
                for (j=1;j<=2;j++)
                {
                         s[i][j]= a[i][j]+b[i][j];
```

```
            }
        }
        printf("The addition matrix is:\n");
        for (i=1;i<=2;i++)
        {
            for (j=1;j<=2;j++)
            {
                printf("%d\n",s[i][j] );
            }
        }
        for(i=1;i<=2;i++)
        {
            for(j=1;j<=2;j++)
            {
                s[i][j]=0;
                for(k=1;k<=2;k++)
                {
                    s[i][j] =s[i][j]+a[i][k]*b[k][j];
                }
            }
        }
        printf("Matrix Multiplication Is: \n");
        for(i=1;i<=2;i++)
        {
            for (j=1;j<=2;j++)
            {
                printf("%d\n",s[i][j]);
            }
        }
}
```

8. Write a program to perform implementation of sparse matrix.

Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[10][10],b[10][3],r,c,s=0,i,j;
        clrscr();
        printf("\nenter the order of the sparse matrix");
        scanf("%d%d",&r,&c);
        printf("\nenter the elements in the sparse matrix(mostly zeroes)");

        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                        printf("\n%d row and %d column ",i,j);
                        scanf("%d",&a[i][j]);
                        if(a[i][j]!=0)
                        {
                                b[s][0]=a[i][j];
                                b[s][1]=i;
                                b[s][2]=j;
                                s++;
                        }
                }

                printf("\nthe sparse matrix is given by");
                printf("\n");
```

```
        for(i=0;i<s;i++)
        {
                for(j=0;j<3;j++)
                {
                `       printf("%d ",b[i][j]);
                }
                printf("\n");
        }
        getch();
}
```

9. Write a program to perform insertion,insertion at position ,deletion ,deletion from a position and displaying a linked list.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node
{
        int info;
        struct node *next;
};
typedef struct node NODE;
NODE *start;
void createmptylist(NODE **start)
{
        *start=(NODE *)NULL;
}
void traversinorder(NODE *start)
{
        while(start != (NODE *) NULL)
        {
                printf("%d\n",start->info);
                start=start->next;
        }
}
void insertatbegin(int item)
{
        NODE *ptr;
        ptr=(NODE *)malloc(sizeof(NODE));
        ptr->info=item;
```

```
        if(start==(NODE *)NULL)
                ptr->next=(NODE *)NULL;
        else
                ptr->next=start;
        start=ptr;
}
void insert_at_end(int item)
{
        NODE *ptr,*loc;
        ptr=(NODE *)malloc(sizeof(NODE));
        ptr->info=item;
        ptr->next=(NODE *)NULL;
        if(start==(NODE*)NULL)
                start=ptr;
        else
        {
                loc=start;
                while(loc->next!=(NODE *)NULL)
                        loc=loc->next;
                loc->next=ptr;
        }
}

void dele_beg(void)
{
        NODE *ptr;
        if(start==(NODE *)NULL)
                return;
        else
        {
                ptr=start;
```

```
                start=(start)->next;
                free(ptr);
        }
}

void dele_end(NODE *start)
{
        NODE *ptr,*loc;
        if(start==(NODE *)NULL)
                return;
        else if((start)->next==(NODE *)NULL)
        {
                ptr=start;
                start=(NODE *)NULL;
                free(ptr);
        }
        else
        {
                loc=start;
                ptr=(start)->next;
                while(ptr->next!=(NODE *)NULL)
                {
                        loc=ptr;
                        ptr=ptr->next;
                }
                loc->next=(NODE *)NULL;
                        free(ptr);
        }
}
```

```
void insert_spe(NODE *start,int item)
{
        NODE *ptr,*loc;
        int temp,k;
        for(k=0,loc=start;k<temp;k++)
        {
                loc=loc->next;
                if(loc==NULL)
                {
                        printf("node in the list at less than one\n");
                        return;
                }
        }
        ptr=(NODE *)malloc(sizeof(NODE));
        ptr->info=item;
        ptr->next=loc->next;;
        loc->next=ptr;
}

dele_spe(NODE *start)
{
        NODE *ptr,*loc;
        int temp;
        printf("\nEnter the which element do you want to delete\n");
        scanf("%d",&temp);
        ptr=start;
        if(start==NULL)
        {
                printf("Empty list\n");
                return;
        }
```

```
        else
        {
                loc=ptr;
                while(ptr!=NULL)
                {
                        if(ptr->info==temp)
                        {
                                loc->next=ptr->next;
                                free(ptr);
                                return;
                        }
                        loc=ptr;
                        ptr=ptr->next;
                }
        }
}

void main()
{
        int choice,item,after;
        char ch;
        clrscr();
        createmptylist(start);
        do
        {
                printf("1.Insert element at begin \n");
                printf("2. insert element at end positon\n");
                printf("3. insert specific the position\n");
                printf("4.travers the list in order\n");
                printf("5. delete from the begin\n");
                printf("6. delete from the last\n");
```

```
printf("7. delete the element from the specific position\n");
printf("8. exit\n");
printf("enter your choice\n");
scanf("%d",&choice);
switch(choice)
{
        case 1:
                printf("Enter the item\n");
                scanf("%d",&item);
                insertatbegin(item);
                break;
        case 2:
                printf("Enter the item\n");
                scanf("%d",&item);
                insert_at_end(item);
                break;
        case 3:
                printf("Enter the item\n");
                scanf("%d",&item);
                insert_spe(start,item);
                break;
        case 4:
                printf("\ntravers the list\n");
                traversinorder(start);
                break;
        case 5:
                printf("Delete the item\n");
                dele_beg();
                break;
```

```
                    case 6:
                            printf("Delete the element\n");
                            dele_end(start);
                            break;
                    case 7:
                            printf("Delete the element");
                            dele_spe(start);
                            break;
                    case 8:
                             return;
                }
                fflush(stdin);
                printf("do your want continous\n");
                scanf("%c",&ch);
        }while((ch='y')||(ch='y'));
        getch();
}
```

10. Write a program to perform implementation of stack

Code:

```
#include<stdio.h>
#include<conio.h>
#define MAXSIZE 10
void push();
int pop();
void traverse();
int stack[MAXSIZE];
int Top=-1;
void main()
{
        int choice;
        char ch;
        do
        {
                clrscr();
                printf("\n1. PUSH ");
                printf("\n2. POP ");
                printf("\n3. TRAVERSE ");
                printf("\nEnter your choice");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: push();
                                break;
                        case 2:printf("\nThe deleted element is %d",pop());
                                break;
                        case 3:  traverse();
                                 break;
```

```
                default: printf("\nYou Entered Wrong Choice");
            }
            printf("\nDo You Wish To Continue (Y/N)");
            fflush(stdin);
            scanf("%c",&ch);
        }while(ch=='Y' || ch=='y');
}

void push()
{
        int item;
        if(Top == MAXSIZE - 1)
        {
            printf("\nThe Stack Is Full");
            getch();
            exit(0);
        }
        else
        {
            printf("Enter the element to be inserted");
            scanf("%d",&item);
            Top= Top+1;
            stack[Top] = item;
        }
}

int pop()
{
        int item;
        if(Top == -1)
        {
```

```
                printf("The stack is Empty");
                getch();
                exit(0);
        }
        else
        {
                item = stack[Top];
                Top = Top-1;
        }
        return(item);
}

void traverse()
{
        int i;
        if(Top == -1)
        {
                printf("The Stack is Empty");
                getch();
                exit(0);
        }
        else
        {
                for(i=Top;i>=0;i--)
                {
                        printf("Traverse the element");
                        printf("\n%d",stack[i]);
                }
        }
}
```

11. Write a program to perform implementation of stack using linked list.

Code:

```
#include<stdio.h>
#include<conio.h>
struct stack
{
        int no;
        struct stack *next;
}
*start=NULL;
typedef struct stack st;
void push();
int pop();
void display();

void main()
{
        char ch;
        int choice,item;
        do
        {
                clrscr();
                printf("\n 1: push");
                printf("\n 2: pop");
                printf("\n 3: display");
                printf("\n Enter your choice");
                scanf("%d",&choice);
                switch (choice)
                {
                        case 1: push();
```

```
                        break;
                case 2: item=pop();
                        printf("The delete element in %d",item);
                        break;
                case 3: display();
                        break;
                default : printf("\n Wrong choice");
            }
            printf("\n do you want to continue(Y/N)");
            fflush(stdin);
            scanf("%c",&ch);
        }while (ch=='Y'||ch=='y');
}

void push()
{
        st *node;
        node=(st *)malloc(sizeof(st));
        printf("\n Enter the number to be insert");
        scanf("%d",&node->no);
        node->next=start;
        start=node;
}

int pop()
{
        st *temp;
        temp=start;
        if(start==NULL)
        {
                printf("stack is already empty");
```

```
                getch();
                exit();
        }
        else
        {
                start=start->next;
                free(temp);
        }
        return(temp->no);
}

void display()
{
        st *temp;
        temp=start;
        while(temp->next!=NULL)
        {
                printf("\nno=%d",temp->no);
                temp=temp->next;
        }
        printf("\nno=%d",temp->no);
}
```

12. Write a program to perform implementation of queue.

Code:

```
#include<stdio.h>
#include<conio.h>
int queue[5];
long front,rear;
void main()
{
        int choice,info;
        clrscr();
        //Initialising queue
        initqueue();
        while(1)
        {
                clrscr();
                 //Displaying menu
                printf("            MENU            \n");
                printf("1. Insert  an element in queue\n");
                printf("2.delete  an element from queue\n");
                printf("3.Display the queue\n");
                printf("4.Exit!\n");
                printf("Your choice: ");
                scanf("%i",&choice);
                switch(choice)
                {
                        case 1:
                                if(rear<4)
                                {
                                        printf("enter the number");
                                        scanf("%d",&info);
```

```
                if (front==-1)
                {
                        front=0;
                        rear=0;
                }
                else
                        rear=rear+1;
                        queue[rear]=info;
         }
        else
                printf("queue is full");
        getch();
        break;
case 2:
        int info;
        if(front!=-1)
        {
                info=queue[front];
                 if(front==rear)
                {
                        front=-1;
                         rear=-1;
                }
                else
                        front=front+1;
                        printf("no deleted is = %d",info);
        }
        else
                printf("queue is empty");
        getch();
        break;
```

```
                        case 3: display();
                                getch();
                                 break;
                        case 4: exit();
                                break;
                        default:
                                printf("You entered wrong choice!");
                                getch();
                                break;
                }
        }
}

void initqueue()
{
         //Initialising front & rear to -1
        front=rear=-1;
}

/*displays the current position of the queue*/
void display()
{
        int i; //For loop driver
        //Displaying elements in queue
        for(i=front;i<=rear;i++)
        printf("%i\n",queue[i]);
}
```

13. Write a program to perform implementation of queue using linked list.

Code :

```
#include<stdio.h>
struct queue
{
        int no;
        struct queue *next;
}
*start=NULL;
void add();
int del();
void traverse();
void main()
{
        int ch;
        char choice;
        do
        {
                printf("----1. add\n");
                printf("----2. delete\n");
                printf("----3. traverse\n");
                printf("----4. exit\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: add();
                                break;
                        case 2: printf("the delete element is\n%d",del());
                                break;
```

```
                        case 3: traverse();
                                break;
                        case 4: return;
                        default : printf("wrong choice\n");
                }
                fflush(stdin);
                scanf("%c",&choice);
        }
        while(choice!=4);
}
void add()
{
        struct queue *p,*temp;
        temp=start;
        p=(struct queue*)malloc(sizeof(struct queue));
        printf("Enter the data");
        scanf("%d",&p->no);
        p->next=NULL;
        if(start==NULL)
        {
                start=p;
        }
        else
        {
                while(temp->next!=NULL)
                {
                        temp=temp->next;
                }
                temp->next=p;
        }
}
```

```
int del()
{
        struct queue *temp;
        int value;
        if(start==NULL)
        {
                printf("queue is empty");
                getch();
                return(0);
        }
`       else
        {
                temp=start;
                value=temp->no;
                start=start->next;
                free(temp);
        }
        return(value);
}
void traverse()
{
        struct queue *temp;
        temp=start;
        while(temp->next!=NULL)
        {
                printf("no=%d",temp->no);
                temp=temp->next;
        }
        printf("no=%d",temp->no);
}
```

14. Write a program to perform binary search.

Code:

```
#include<stdio.h>
 main()
{
        int c, first, last, middle, n, search, array[100];
         printf("Enter number of elements\n");
        scanf("%d",&n);
         printf("Enter %d integers\n", n);
         for ( c = 0 ; c < n ; c++ )
        {
                 scanf("%d",&array[c]);
                printf("Enter value to find\n");
                 scanf("%d",&search);
                first = 0;
                 last = n - 1;
                 middle = (first+last)/2;
                 while( first <= last )
                 {
                        if ( array[middle] < search )
                                first = middle + 1;
                         else if ( array[middle] == search )
                         {
                                printf("%d found at location %d.\n", search,
                                middle+1);
                                break;
                        }
                        else
                        {
                                 last = middle - 1;
```

```
                middle = (first + last)/2;
            }
            if ( first > last )
                printf("Not found! %d is not present in the list.\n",
                search);
            return 0;
        }
    }
```

15. Write a program to implement doubly linked list.

Code:

```
# include <stdio.h>
# include <malloc.h>
struct Dlink_list
{
        struct Dlink_list *previous;
        int info;
        struct Dlink_list *next;
};
typedef struct Dlink_list Dnode;
/* Global declaration */
int num ;
Dnode *lp=NULL,*rp=NULL;
/* Prototypes of various functions */
void Create(Dnode *);
void display (Dnode *);

/* Function main */
void main()
{
        lp = (Dnode *) malloc(sizeof(Dnode));
        Create(lp);
        display(lp);
        getch();
}

/* Function create */
void Create(Dnode *ptr)
{
```

```
        char ch;
        clrscr();
        num = 0;
        lp->previous = NULL;
        printf("\n Enter E for Exit any other key for continue: ");
        ch = getchar();
        if(ch=='e')
        {
                free(lp);
                exit(0);
        }
        do
        {
                printf("\n Input the values of the node : %d: ", (num+1));
                scanf("%d", &ptr->info);
                fflush(stdin);
                printf("\n want to continue(y/n)->: ");
                ch = getchar();
                if(ch=='y')
                {
                        ptr->next = (Dnode *) malloc(sizeof(Dnode));
                        ptr->next->previous = ptr;
                        ptr = ptr->next;
                }
                num ++;
        }while(ch=='y');
        ptr->next=NULL;
        rp=ptr; /* Assign the adress of rightmost node to rp */
        printf("\n Total nodes = %d", num);
}
```

```
/* DELETE LAST NODE FROM A SIMPLE DOUBLY LINKED LIST*/
/* Function delete */
void Delete (Dnode *ptr)
{
      if ( lp == NULL)   /* for empty list */
      {
            printf("\n List is empty\n");
            exit(0);
      }
      if(lp==rp)    /* for list containing single node */
      {
            lp=rp=NULL;
            free(ptr);
      }
      else            /* for list containing more than one node */
      {
            ptr=rp;
            rp=ptr->previous;
            rp->next=NULL;
            free(ptr);
      }
}

/* DELETE FIRST NODE FROM A SIMPLE DOUBLY LINKED LIST */
void Delete (Dnode *ptr)
{
      if ( ptr == NULL)
      {
            printf("\n List is empty\n");
            exit(0);
      }
```

```
        if(lp==rp)
        {
                lp=rp=NULL;
                free(ptr);
        }
        else
        {
                lp=ptr->next;
                lp->previous=NULL;
                free(ptr);
        }
}

/* Functio Display  */
void display (Dnode *ptr)
{
        while(ptr!=NULL)  /* Traverse up to end of the list */
        {
                printf("\n 0x%x", ptr);
                printf(" %d", ptr->info);
                ptr = ptr->next;
        }
}
```

16. Write a program to perform implementation of polynomial expression and perform addition of two polynomial expressions.

Code :

```
#include<stdio.h>
#include<conio.h>
#include<limits.h>
int select();
struct rec
`{
        float coef;
        int exp;
        struct rec *next;
};
struct rec *rear;
struct rec *create(struct rec *list);
void *add(struct rec *first,struct rec *second);
struct rec *insert(double coef,int exp,struct rec *rear);
void *display(struct rec *list);
int nodes;

void main()
{
        struct rec *first=NULL,*second=NULL;
        int choice;
        do
        {
                choice=select();
                switch(choice)
                {
                        case 1:
```

```
                        first=create(first);continue;
                case 2:
                        second=create(second);continue;
                case 3:
                        add(first,second);continue;
                case 4:
                        puts("END");exit(0);
                }
        }while(choice!=4);
}

int select()
{
        int selection;
        do
        {
                puts("Enter 1: create the first list");
                puts("Enter 2: create the second list");
                puts("Enter 3: add the two list");
                puts("Enter 4: END");
                puts("Entr your choice");
                scanf("%d",&selection);
        }while((selection<1)||(selection>4));
        return (selection);
}

struct rec *create(struct rec *x)
{
        float coef;
        int exp;
        int endexp=INT_MAX;
```

```
        struct rec *element;
        puts("Enter cofs &exp:exp in descending order:""to quit enter 0 for exp");
        x=(struct rec *)malloc(sizeof(struct rec));
        x->next=NULL;
        rear=x;
        for(;;)
        {
                puts("Enter coefficient");
                element=(struct rec*)malloc(sizeof(struct rec));
                scanf("%f",&coef);
                element->coef=coef;
                if(element->coef==0.0)break;
                puts("Enter exponent");
                scanf("%d",&exp);
                element->exp=exp;
                if((element->exp<=0)||(element->exp>=endexp))
                {
                        puts("Invalid exponent");
                        break;
                }
                element->next=NULL;
                rear->next=element;
                rear=element;
        }
        x=x->next;
        return(x);
}

void *add(struct rec *first,struct rec *second)
{
        float total;
```

```
struct rec *end,*rear,*result;
result=(struct rec *)malloc(sizeof(struct rec));
rear=end;
while((first!=NULL)&&(second!=NULL))
{
        if(first->exp==second->exp)
        {
                if((total=first->exp+second->exp)!=0.0)
                rear=insert(total,first->exp,rear);
                first=first->next;
                second=second->next;
        }
        else
                if(first->exp>second->exp)
                {
                        rear=insert(first->coef,first->exp,rear);
                        first=first->next;
                }
                else
                {
                        rear=insert(second->coef,second->exp,rear);
                        second=second->next;
                }
        }
        for(;first;first=first->next)
                rear=insert(first->coef,first->exp,rear);
        for(;second;second=second->next)
                rear=insert(second->coef,second->exp,rear);
        rear->next=NULL;
        display(end->next);
        free(end);
```

```
}

void *display(struct rec *head)
{
        while(head!=NULL)
        {
                printf("%2lf",head->coef);
                printf("%2d",head->exp);
                head=head->next;
        }
        printf("\n");
}

struct rec *insert(double coef,int exp,struct rec *rear)
{
        rear->next=(struct rec *)malloc(sizeof(struct rec));
        rear=rear->next;
        rear->coef=coef;
        rear->exp=exp;
        return(rear);
}
```

Top 50 Data Structures Interview Questions & Answers

1) What is data structure?

Data structure refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with the data structure, we not only focus on one piece of data but the different set of data and how they can relate to one another in an organized manner.

2) Differentiate between file and structure storage structure.

The key difference between both the data structure is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

3) When is a binary search best applied?

A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is searched starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

4) What is a linked list?

A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link for data storage.

5) How do you reference all the elements in a one-dimension array?

To reference all the elements in a one -dimension array, you need to use an indexed loop, So that, the counter runs from 0 to the array size minus one. In this manner, You can reference all

the elements in sequence by using the loop counter as the array subscript.

6) In what areas do data structures are applied?

Data structures are essential in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

7) What is LIFO?

LIFO is a short form of Last In First Out. It refers how data is accessed, stored and retrieved. Using this scheme, data that was stored last should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

8) What is a queue?

A queue is a data structure that can simulate a list or stream of data. In this structure, new elements are inserted at one end, and existing elements are removed from the other end.

9) What are binary trees?

A binary tree is one type of data structure that has two nodes, a left node, and a right node. In programming, binary trees are an extension of the linked list structures.

10) Which data structures are applied when dealing with a recursive function?

Recursion, is a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?

A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree

A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?

Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?

It depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?

Apart from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?

FIFO stands for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?

An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?

Merge sort, is a divide-and-conquer approach for sorting the data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

19) Differentiate NULL and VOID

Null is a value, whereas Void is a data type identifier. A variable that is given a Null value indicates an empty value. The void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?

A linked list is an ideal data structure because it can be modified easily. This means that editing a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?

Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes

data being added to it, meaning data is being “pushed” into the stack. On the other hand, a pop denotes data retrieval, and in particular, refers to the topmost data being accessed.

22) What is a linear search?

A linear search refers to the way a target key is being searched in a sequential data structure. In this method, each element in the list is checked and compared against the target key. The process is repeated until found or if the end of the file has been reached.

23) How does variable declaration affect memory allocation?

The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?

The heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?

A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is Data abstraction?

Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

27) How do you insert a new item in a binary search tree?

Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

28) How does a selection sort work for an array?

The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. In this process, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position.

The smallest element remaining in the subarray is then located next to subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

29) How do signed and unsigned numbers affect memory?

In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (an unsigned 8-bit number has a range 0-255, while the 8-bit signed number has a range -128 to +127.

30) What is the minimum number of nodes that a binary tree can have?

A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

31) What are dynamic data structures?

Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

32) In what data structures are pointers applied?

Pointers that are used in linked list have various applications in the data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

33) Do all declaration statements result in a fixed reservation in memory?

Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

34) What are ARRAYs?

When dealing with arrays, data is stored and retrieved using an index that refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

35) What is the minimum number of queues needed when implementing a priority queue?

The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is used for actual storage of data.

36) Which sorting algorithm is considered the fastest?

There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort,

merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

37) Differentiate STACK from ARRAY.

Stack follows a LIFO pattern. It means that data access follows a sequence wherein the last data to be stored when the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

38) Give a basic algorithm for searching a binary search tree.

1. if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if a target is not in the root item, check if a target is smaller than the root's value
5. if a target is smaller than the root's value, search the left subtree
6. else, search the right subtree

39) What is a dequeue?

A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

40) What is a bubble sort and how do you perform it?

A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values “bubble” to the top of the list, while the larger value sinks to the bottom.

41) What are the parts of a linked list?

A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes. All these nodes are linked sequentially.

42) How does selection sort work?

Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

43) What is a graph?

A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs

are also referred to as edges or arcs and are used to connect nodes where data can be stored and retrieved.

44) Differentiate linear from a nonlinear data structure.

The linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks, and queues. On the other hand, a non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of nonlinear data structure include trees and graphs.

45) What is an AVL tree?

An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

46) What are doubly linked lists?

Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and another one that connects to the previous node.

47) What is Huffman's algorithm?

Huffman's algorithm is used for creating extended binary trees that have minimum weighted path lengths from the given weights. It makes use of a table that contains the frequency of occurrence for each data element.

48) What is Fibonacci search?

Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can significantly reduce the time needed in order to reach the target element.

49) Briefly explain recursive algorithm.

Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

50) How do you search for a target key in a linked list?

To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is

reached.