**NEPAL ENGINEERING COLLEGE**

**DATA STRUCTURES AND ALGORITHMS LABSHEET**

**Arrays**

1. Write a program using user defined functions to find sum of 2- dimensional arrays (m\*n Matrix) and store the sum of corresponding elements into third array and make a print function to print values of the matrix at the corresponding places (Not in a single row), get the transpose of the third matrix and call print function to display all 4matrices.
2. Write a program to multiply two m\*n matrices using dynamic memory allocation. Hints: 2 matrices are multipliable only if the columns of first matrix are equal to rows of second matrix. The resultant matrix will be of the size r1\*c2. Use three nested for loops and use the following expression to calculate the element of third matrix:

C[i][j]+=a[i][k]\*b[k][j]

**ADT**

1. Create a ADT point and using that data type find the distance between two points.

**Stack**

1. Write a menu driven program to illustrate basic operation of stack using array:
2. Push
3. Pop
4. Top
5. Display all

#include<stdio.h>

#define max 5

#include<process.h>

struct stack

{

int array[max];

int tos;

};

void push(struct stack \*s, int item)

{

if(s->tos==max-1)

{

printf("Stack overflow");

}

else

{

s->array[++s->tos]=item;

}

}

int pop(struct stack \*s)

{

if(s->tos==-1)

{

printf("stack underflow");

}

return(s->array[s->tos--]);

}

int top(struct stack \*s)

{

return (s->array[s->tos]);

}

void display(struct stack \*s)

{

int i;

for( i=0;i<=s->tos;i++)

{

printf("%d\t",s->array[i]);

}

}

int main()

{

int ch,item;

struct stack s1;

s1.tos=-1;

while(1)

{

printf("\n1.push\n2.pop\n3.display\n4.exit\nenter your choice");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("enter the item to push in stack");

scanf("%d",&item);

push(&s1,item);

break;

case 2:

printf("\nthe deleted element is %d",pop(&s1));

break;

case 3:

display(&s1);

break;

case 4:

exit(0);

}

}

return 0;

}

1. Write a program to evaluate a postfix expression.

#include<stdio.h>

#include<math.h>

int stack[10],top;

int pop()

{

return stack[top--];

}

void push(int data)

{

top++;

stack[top]=data;

}

void evaluate(char infix[25])

{

char se;

int i=0,data,op1,op2;

while(infix[i]!='\0')

{

se=infix[i];

switch(se)

{

case '+':op2=pop();op1=pop();push(op2+op1);break;

case '-':op2=pop();op1=pop();push(op2-op1);break;

case '\*':op2=pop();op1=pop();push(op2\*op1);break;

case '/':op2=pop();op1=pop();push(op2/op1);break;

case '^':op2=pop();op1=pop();push(pow(op2,op1));break;

default:printf("enter the value of %c",se);

scanf("%d",&data);

push(data);

}

i++;

}

}

int main()

{

char infix[25];

printf("enter the infix expression");

gets(infix);

evaluate(infix);

printf("the evaluated expression is %d",stack[top]);

}

1. Write a program to convert Infix Expression into Postfix Expression.

#include<stdio.h>

#include<string.h>

char stack[25];

int top=-1;

void push(char item)

{

top++;

stack[top]=item;

}

char pop()

{

if(top==-1)

{

exit(0);

}

return stack[top--];

}

int priority(char symbol)

{

if(symbol==94)

{

return 3;

}

if(symbol==47||symbol==42)

{

return 2;

}

else if(symbol=43||symbol==45)

{

return 1;

}

else

{

return 0;

}

}

void convert(char infix[25])

{

int index=0;

char scan\_element,temp;

while(infix[index]!='\0')

{

scan\_element=infix[index];

switch(scan\_element)

{

case '(':

push(scan\_element);

break;

case')':

temp=pop();

while(temp!='(')

{

printf("%c",temp);

temp=pop();

}

break;

case'+':

case'-':

case'\*':

case'/':

case'^':

while(priority(stack[top])>=priority(scan\_element))

{

temp=pop();

printf("%c",temp);

}

push(scan\_element);

break;

default: printf("%c",scan\_element);

break;

}

index++;

}

while(top!=-1)

{

if(stack[top]='(')

{

}

else

printf("%c",pop());

}

}

int main()

{

char infix[25];

stack[-1]="(";

printf("enter the infix expression");

gets(infix);

convert(infix);

}

**Queue**

1. Write a menu driven program to illustrate basic operations of linear Queue:
2. Enqueue
3. Dequeue
4. Display all values

**Strategy 1**

#include<process.h>

# define max 10

struct queue

{

int q[max];

int front;

int rear;

};

void enqueue(int data,struct queue \*q1)

{

//printf("\nF:%d R:%d\n",q1->front,q1->rear);

int i,j;

if(q1->front==0&&q1->rear==max-1)

{

printf("overflow");

return;

}

if(q1->rear==max-1)

{

i=0;j=q1->front;

while(j<=q1->rear)

{

q1->q[i]=q1->q[j];

i++;

j++;

}

q1->rear=q1->rear-q1->front;

q1->front =0;

}

q1->rear++;

q1->q[q1->rear]=data;

//printf("\nF:%d R:%d\n",q1->front,q1->rear);

}

int dequeue(struct queue \*q1)

{

//printf("\nF:%d R:%d\n",q1->front,q1->rear);

if(q1->rear<q1->front)

{

printf("underflow");

return 0;

}

return(q1->q[q1->front++]);

}

main()

{

struct queue q1;

int ch,data,i;

q1.front=0;

q1.rear=-1;

while(1)

{

printf("\n 1.enqueue \n 2. dequeue \n 3.display \n 4.exit\n ENTER YOUR CHOICE:");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("enter the data to enqueue");

scanf("%d",&data);

enqueue(data,&q1);

break;

case 2:

printf("the dequed item is %d",dequeue(&q1));

break;

case 3:

for(i=q1.front;i<=q1.rear;i++)

{

printf("\t%d",q1.q[i]);

}

break;

case 4:

exit(0);

}

}

}

**Strategy** **2**

#include<stdio.h>

#include<process.h>

#include<dos.h>

# define max 5

struct queue

{

int q[max];

int front;

int rear;

};

void enqueue(int data,struct queue \*q1)

{

if(q1->rear==max-1)

{

printf("overflow");

return;

}

q1->rear++;

q1->q[q1->rear]=data;

//printf("\nF:%d R:%d\n",q1->front,q1->rear);

}

int dequeue(struct queue \*q1)

{

int data,i=0;

//printf("\nF:%d R:%d\n",q1->front,q1->rear);

if(q1->rear==-1)

{

printf("underflow");

return 0;

}

//shifting

data=q1->q[0];

while(i<q1->rear)

{

q1->q[i]=q1->q[i+1];

i++;

}

q1->rear--;

return data;

}

main()

{

struct queue q1;

int ch,data,i;

q1.front=0;

q1.rear=-1;

while(1)

{

system("cls");

printf("\n 1.enqueue \n 2. dequeue \n 3.display \n 4.exit\n ENTER YOUR CHOICE:");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("enter the data to enqueue");

scanf("%d",&data);

enqueue(data,&q1);

break;

case 2:

printf("the dequed item is %d",dequeue(&q1));

getch();

break;

case 3:

printf("the items in queue are\n");

for(i=0;i<=q1.rear;i++)

{

printf("\t%d",q1.q[i]);

}

getch();

break;

case 4:

exit(0);

}

}

}

1. Write a menu driven program to illustrate basic operations of Circular queue:
2. Using counter

// this is a simple strategy and should be done on your own

1. With one position vacant

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

# define max 5

int cqueue[max],front=max-1,rear=max-1;

void enqueue()

{

int item;

if(front==(rear+1)%max)

{

printf("overflow");

return;

}

printf("enter the data to insert");

scanf("%d",&item);

rear=(rear+1)%max;

cqueue[rear]=item;

}

void dequeue()

{

if(front==rear)

{

printf("underflow");

}

else

{

front=(front+1)%max;

printf("%d",cqueue[front]);

}

}

void display()

{

int i;

if (rear >= front)

{

for (i = front+1; i <= rear; i++)

printf("%d ",cqueue[i]);

}

else

{

for ( i = front; i < max-1; i++)

printf("%d ", cqueue[i]);

for (i = 0; i <= rear; i++)

printf("%d ", cqueue[i]);

}

}

int main()

{

int ch;

while(1)

{

printf("\n1.enqueue\n2.dequeue\n3.display\n4.exit\n enter your choice");

scanf("%d",&ch);

switch(ch)

{

case 1: enqueue();break;

case 2: dequeue();break;

case 3: display();break;

case 4: exit(0);break;

}

}

}

**Linked List**

1. Write a program using an Array to perform the following tasks for contiguous list: (Use Switch

case for menu)

1. Insert: Insert a value at the end / at any specified position.
2. Delete: Delete a value from the specified position
3. Traverse: Print all the elements from 0 to last position
4. Searching: Searching a particular value in the array

/**/ Contiguous list means array, this problem should be done by you on your own**

1. Write a menu driven program to illustrate basic operations of Singly Linked List with following operations
2. Insert at first
3. Insert at last
4. Insert at nth position
5. Delete from first
6. Delete from last
7. Delete from nth position
8. Traverse all the nodes
9. Search any value

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int info;

struct node \*link;

}\*start;

void main()

{

int ch,n,m,i;

clrscr();

start=NULL;

printf(“%x--> %d”,start->link,start->info);

while(1)

{

printf("\n1:Create list");

printf("\n2:Add at begining");

printf("\n3:Add at last");

printf("\n4:Delete from begining");

printf("\n5:Delete from last");

printf("\n6:Count");

printf("\n7:Search");

printf("\n8:Display");

printf("\n9:Add after");

printf("\n10:Delete particular node");

printf("\n11:Reserve");

printf("\n12:Quit");

printf("\n13:Sorting");

printf("\n\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\nHow many nodes you want:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("\nEnter the %d element:",i+1);

scanf("%d",&m);

create(m);

}

break;

case 2:

printf("\nEnter the element:");

scanf("%d",&m);

add\_beg(m);

break;

case 3:

printf("\nEnter the element:");

scanf("%d",&m);

add\_end(m);

break;

case 4:

del\_beg();

break;

case 5:

del\_end();

break;

case 6:

count();

break;

case 7:

printf("\nEnter the element you want to search:");

scanf("%d",&m);

search(m);

break;

case 8:

display();

break;

case 9:

printf("\nEnter the node after which you want to add");

scanf("%d",&n);

printf("\nEnter value");

scanf("%d",&m);

addafter(n,m);

break;

case 10:

printf("\nEnter the node you want to delete:");

scanf("%d",&m);

del(m);

break;

case 11:

rev();

break;

case 12:

exit(0);

case 13:

sort();

break;

default:

printf("\nYou have wrong choice:");

} /\* End of switch \*/

} /\* End of while \*/

} /\* End of main \*/

create(int data)

{

struct node \*q,\*tmp;

tmp=malloc(sizeof(struct node));

tmp->info=data;

tmp->link=NULL;

if(start==NULL)

start=tmp;

else

{

q=start;

while(q->link!=NULL)

q=q->link;

q->link=tmp;

}

}

add\_beg(int data)

{

struct node \*tmp;

tmp=malloc(sizeof(struct node));

tmp->info=data;

tmp->link=start;

start=tmp;

}

add\_end(int data)

{

struct node \*q, \*tmp;

q=start;

tmp=malloc(sizeof(struct node));

tmp->info=data;

while(q->link!=NULL)

q=q->link;

q->link=tmp;

tmp->link=NULL;

}

display()

{

struct node \*q;

if(start==NULL)

printf("\nList is empty:");

else

{

q=start;

printf("\nThe elements are:");

while(q!=NULL)

{

printf("\n%d",q->info);

q=q->link;

}

}

}

del\_beg()

{

struct node \*q;

if(start==NULL)

printf("\nList is empty , deletion not possible:");

else

q=start;

start=start->link;

free(q);

}

del\_end()

{

struct node \*q,\*tmp;

if(start==NULL)

printf("\nList is empty, deletion not possible:");

else

q=start;

while(q->link->link!=NULL)

{

tmp=q->link->link;

q=q->link;

}

free(tmp);

q->link=NULL;

}

count()

{

struct node \*q;

int i=1;

if(start==NULL)

printf("\nList is empty:");

else

{

q=start;

while(q->link!=NULL)

{

i++;

q=q->link;

}

printf("\nList has %d nodes:",i);

}

return i;

}

search(int data)

{

struct node \*q;

int f=0;

q=start;

while(q->link!=NULL)

{

if(q->info==data)

f=1;

q=q->link;

}

if(f==1)

printf("\nItem is present:");

else

printf("\nItem is not present:");

}

addafter(int data1,int data2)

{

struct node \*tmp,\*q;

tmp=malloc(sizeof(struct node));

q=start;

while(q->link!=NULL)

{

if(q->info==data1)

{

tmp->info=data2;

tmp->link=q->link;

q->link=tmp;

}

q=q->link;

}

}

del(int data)

{

struct node \*q,\*tmp;

q=start;

while(q->link->link!=NULL)

{

if(q->link->info==data)

{

tmp=q->link;

q->link=tmp->link;

free(tmp);

}

q=q->link;

}

}

rev()

{

struct node \*p1,\*p2,\*p3;

if(start->link==NULL)

return;

p1=start;

p2=p1->link;

p3=p2->link;

p1->link=NULL;

p2->link=p1;

while(p3!=NULL)

{

p1=p2;

p2=p3;

p3=p3->link;

p2->link=p1;

}

start=p2;

}

sort()

{

struct node \*tmp,\*q;

int i,j,n;

n=count();

for(i=0;i<n;i++)

{

q=start;

for(j=1;j<n;j++)

{

if(q->info>q->link->info)

{

tmp=q->info;

q->info=q->link->info;

q->link->info=tmp;

}

q=q->link;

}

}

}

1. Write a menu driven program to illustrate basic operations of Stack using linked list.

#include<stdio.h>

#include<conio.h>

struct Node

{

int data;

struct Node \*next;

}\*top = NULL;

void push(int);

void pop();

void display();

void main()

{

int choice, value;

clrscr();

printf("\n:: Stack using Linked List ::\n");

while(1){

printf("\n\*\*\*\*\*\* MENU \*\*\*\*\*\*\n");

printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d",&choice);

switch(choice){

case 1: printf("Enter the value to be insert: ");

scanf("%d", &value);

push(value);

break;

case 2: pop(); break;

case 3: display(); break;

case 4: exit(0);

default: printf("\nWrong selection!!! Please try again!!!\n");

}

}

}

void push(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

if(top == NULL)

newNode->next = NULL;

else

newNode->next = top;

top = newNode;

printf("\nInsertion is Success!!!\n");

}

void pop()

{

if(top == NULL)

printf("\nStack is Empty!!!\n");

else{

struct Node \*temp = top;

printf("\nDeleted element: %d", temp->data);

top = temp->next;

free(temp);

}

}

void display()

{

if(top == NULL)

printf("\nStack is Empty!!!\n");

else{

struct Node \*temp = top;

while(temp->next != NULL){

printf("%d--->",temp->data);

temp = temp -> next;

}

printf("%d--->NULL",temp->data);

}

}

1. Write a menu driven program to illustrate basic operations of queue using linked list.

#include<stdio.h>

#include<conio.h>

struct Node

{

int data;

struct Node \*next;

}\*front = NULL,\*rear = NULL;

void insert(int);

void delete();

void display();

void main()

{

int choice, value;

clrscr();

printf("\n:: Queue Implementation using Linked List ::\n");

while(1){

printf("\n\*\*\*\*\*\* MENU \*\*\*\*\*\*\n");

printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d",&choice);

switch(choice){

case 1: printf("Enter the value to be insert: ");

scanf("%d", &value);

insert(value);

break;

case 2: delete(); break;

case 3: display(); break;

case 4: exit(0);

default: printf("\nWrong selection!!! Please try again!!!\n");

}

}

}

void insert(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode -> next = NULL;

if(front == NULL)

front = rear = newNode;

else{

rear -> next = newNode;

rear = newNode;

}

printf("\nInsertion is Success!!!\n");

}

void delete()

{

if(front == NULL)

printf("\nQueue is Empty!!!\n");

else{

struct Node \*temp = front;

front = front -> next;

printf("\nDeleted element: %d\n", temp->data);

free(temp);

}

}

void display()

{

if(front == NULL)

printf("\nQueue is Empty!!!\n");

else{

struct Node \*temp = front;

while(temp->next != NULL){

printf("%d--->",temp->data);

temp = temp -> next;

}

printf("%d--->NULL\n",temp->data);

}

}

**Sorting**

1. Write a program to implement Bubble Sort algorithm.

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

    int temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

// A function to implement bubble sort

void bubbleSort(int arr[], int n)

{

   int i, j;

   for (i = 0; i < n-1; i++)

       // Last i elements are already in place

       for (j = 0; j < n-i-1; j++)

           if (arr[j] > arr[j+1])

              swap(&arr[j], &arr[j+1]);

}

/\* Function to print an array \*/

void printArray(int arr[], int size)

{

    int i;

    for (i=0; i < size; i++)

        printf("%d ", arr[i]);

    printf("n");

}

// Driver program to test above functions

int main()

{

    int arr[] = {64, 34, 25, 12, 22, 11, 90};

    int n = sizeof(arr)/sizeof(arr[0]);

    bubbleSort(arr, n);

    printf("Sorted array: \n");

    printArray(arr, n);

    return 0;

}

1. Write a program to implement Insertion Sort algorithm.

#include <stdio.h>

#include <math.h>

/\* Function to sort an array using insertion sort\*/

void insertionSort(int arr[], int n)

{

   int i, key, j;

   for (i = 1; i < n; i++)

   {

       key = arr[i];

       j = i-1;

       /\* Move elements of arr[0..i-1], that are

          greater than key, to one position ahead

          of their current position \*/

       while (j >= 0 && arr[j] > key)

       {

           arr[j+1] = arr[j];

           j = j-1;

       }

       arr[j+1] = key;

   }

}

// A utility function to print an array of size n

void printArray(int arr[], int n)

{

   int i;

   for (i=0; i < n; i++)

       printf("%d ", arr[i]);

   printf("\n");

}

/\* Driver program to test insertion sort \*/

int main()

{

    int arr[] = {12, 11, 13, 5, 6};

    int n = sizeof(arr)/sizeof(arr[0]);

    insertionSort(arr, n);

    printArray(arr, n);

    return 0;

}

1. Write a program to implement Selection Sort algorithm.

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

    int temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

void selectionSort(int arr[], int n)

{

    int i, j, min\_idx;

    // One by one move boundary of unsorted subarray

    for (i = 0; i < n-1; i++)

    {

        // Find the minimum element in unsorted array

        min\_idx = i;

        for (j = i+1; j < n; j++)

          if (arr[j] < arr[min\_idx])

            min\_idx = j;

        // Swap the found minimum element with the first element

        swap(&arr[min\_idx], &arr[i]);

    }

}

/\* Function to print an array \*/

void printArray(int arr[], int size)

{

    int i;

    for (i=0; i < size; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

// Driver program to test above functions

int main()

{

    int arr[] = {64, 25, 12, 22, 11};

    int n = sizeof(arr)/sizeof(arr[0]);

    selectionSort(arr, n);

    printf("Sorted array: \n");

    printArray(arr, n);

    return 0;

}

**Searching**

1. Write a program to implement linear search algorithm.

#include<stdio.h>

#include<conio.h>

void main()

{

clrscr();

int arr[10], i, num, n, c=0, pos;

printf("Enter the array size : ");

scanf("%d",&n);

printf("Enter Array Elements : ");

for(i=0; i<n; i++)

{

scanf("%d",&arr[i]);

}

printf("Enter the number to be search : ");

scanf("%d",&num);

for(i=0; i<n; i++)

{

if(arr[i]==num)

{

c=1;

pos=i+1;

break;

}

}

if(c==0)

{

printf("Number not found..!!");

}

else

{

printf("%d found at position %d",num, pos);

}

getch();

}

1. Write a program to implement Binary search algorithm.

#include<stdio.h>

#include<conio.h>

void main()

{

clrscr();

int n, i, arr[50], search, first, last, middle;

printf("Enter total number of elements :");

scanf("%d",&n);

printf("Enter %d number :", n);

for (i=0; i<n; i++)

{

scanf("%d",&arr[i]);

}

printf("Enter a number to find :");

scanf("%d", &search);

first = 0;

last = n-1;

middle = (first+last)/2;

while (first <= last)

{

if(arr[middle] < search)

{

first = middle + 1;

}

else if(arr[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.",search);

}

getch();

}