5.Binary search

Algorithm:

Step 1 : Start
Step 2 : print "Enter number of elements of array:"
Step 3 : Read n
Step 4 : print "Enter the array elements:"
Step 5 : Let i=0
Step 6 : Repeat the steps 7-8 until i<n
Step 7 : Read a[i]
Step 8 : Set i=i+1
Step 9 : print "Enter the item to be searched:\n"
Step 10 : Read item
Step 11 : Let beg=0 , end=n-1
Step 12 : Set mid= (beg+end)/2
Step 13 : Repeat the steps 14-17 while beg<=end AND a[mid]<>item
Step 14 : if item< a[mid] then goto step 15 else goto step 16.
Step 15 : end = mid-1
Step 16 : Set beg = mid+1
Step 17 : Set mid= (beg + end)/2
Step 18 : if a[mid]==item then goto step 19 else goto step 20
Step 19 : print "Item found at position" mid+1
Step 20 : print "Item not found:"
Step 21 : Stop.

Output.

Enter number of elements of array:
5
Enter array elements:
50

58

70

88

100

Enter the item to be searched:

70.

Item found at position 3

sort a singly ll

Algorithm.

Step1 : Start
Step2 : Define a structure node that contain data and a link pointer of structure node.
Step3 : Declare struct node pointer head, tail and initialise tail with NULL.
Step4 :

Void addNode ()
Step1 : start
Step2 : Declare structNode pointer newnode and allocate memory space needed for structure node and store the address in newnode.
Step3 : Let newNode → data = data
Step4 : Let newnode → next = NULL
Step5 : If head == NULL then goto step6 else goto step
Step6 : set head = newnode
Step7 : set tail = Neeonode.
Step8 : tail → next = newnode.
Step9 : tail = newNode.
Step10 : Stop.

Void sort (list ().

Step1 : Declare structNode pointer current and initialise it with head and declare another pointer index and initialize it with NULL.
Step2 : Repeat the steps 3 - 10 while current <> NULL
Step3 : index = current → next
Step4 : Repeat the steps 5 - 9. while index <> NULL

Step5 : if current → data > index → data then goto Step 6 else goto step 9
Step6 : temp = current → data.
Step7 : Set current → data = index → data.
Step8 : Set index → data = temp
Step9 : Set current =
Step9 : set index = index → next
Step10 : set current = current → next
Step 11 : stop

void display()

Step1 : start
Step2 : Repeat the steps 3-4 while current <> NULL
Step3 : print current → data
Step4 : set current = current → next
Step5 : stop.

int main().

Step1 : Start
Step2 : addNode (9)
Step3 : addNode (7)
Step4 : addNode (2)
Step5 : addNode (5)
Step6 : addNode (4)
Step7 : print "Original List:"
Step8 : display ().
Step9 : Sort List ().
Step10 : Print ("Sorted List:")
Step11 : display()
Step12 : stop.

Output:

original list:
9 7 2 5 4

sorted list:
2 4 5 7 9

12.Addition of polynomial

Algorithm.

Step1 : Start
Step2 : Set i=0, m=0, k=0
Step3 : Print "Enter the highest degree of polynomial 1:"
Step4 : Read deg1
Step5 : Set i=0
Step6 : Repeat the step. 7-10 until i<=deg1
Step7 : print "Enter the co-efficient of x^" i
Step8 : Read a[i].coeff
Step9 : Set a[k++].expo=i
Step10 : Set i=i+1
Step11 : print "Highest degree of polynomial 2:"
Step12 : Read deg2
Step13 : Set i=0
Step14 : Repeat the steps 15-18 until i<=deg2
Step15 : Print "Enter the coefficient of x^" i
Step16 : Read b[i].coeff
Step17 : b[l++].expo=i
Step18 : Set i=i+1
Step19 : print "Expression 1 =" a[0].coeff
Step20 : Set i=1
Step21 : Repeat the steps 22-23 until i<=deg1
Step22 : print "+"a[i].coeff "x^"a[i].expo
Step23 : Set i=i+1
Step24 : print "Expression 2 =" b[0].coeff.
Step25 : Set i=1
Step26 : Repeat the steps 27-28 until i<=deg2.
Step27 : print "+"b[i].coeff "x^" b[i].expo
Step28 : Set i=i+1
Step29 : If deg1<deg2 then goto step30 else goto step 42.

Step30 : Set i=0
Step31 : Repeat the steps 32- 35. until i <= deg2
Step32 : Set c[m].coeff = a[m].coeff + b[i].coeff
Step33 : Set c[m].expo = a[i].expo
Step34 : Set m=m+1
Step35 : Set i=i+1
Step36 : Set i = deg2+1
Step37 : Repeat the steps 38-41 until i <=deg1
Step38 : set c[m].coeff = a[i].coeff
Step39 : Set c[m].expo = a[i].expo
Step40 : Set m=m+1
Step41 : Set i=i+1
Step42 : Set i=0.
Step43 : Repeat the steps 44-47 until i <= deg1
Step44 : c[m].coeff = a[i].coeff + b[i].coeff
Step45 : Set c[m].expo = b[i].expo.
Step46 : Set m=m+1
Step47 : Set i=i+1
Step48 : Set i=deg1+1
Step49 : Repeat the steps 50-53 until i <=deg2
Step50 : Set c[m].coeff = b[i].coeff
Step51 : set c[m].expo = b[i].expo.
Step52 : Set m=m+1
Step53 : Set i=i+1
Step54 : print " Expression after addition = "c[0].coeff
Step55 : Set i=1
Step56 : Repeat the steps 57-59 until i <=m
Step57 : print "+" c[m].coeff "x^" c[m].expo.
Step58 : Set i=i+1
Step59 : Stop

output.

Enter the highest degree of polynomial 1:
3
Enter the coefficient of x^0 : 1
Enter the coefficient of x^1 : 5
Enter the coefficient of x^2 : 2.
Enter the coefficient of x^3 : 3.

Enter the highest degree of polynomial 2:
2.
Enter the coefficient of x^0 : 3
Enter the coefficient of x^1 : 2.
Enter the coefficient of x^2 : 5

Expression 1 = 1 + 5x^1 + 2x^2 + 3x^3
Expression 2 = 3 + 2x^1 + 5x^2
Expression after addition = 4 + 7x^1 + 7x^2 + 3x^3.

stack using linked l

Algorithm.

Step1 : Start.
Step2 : Create a structure node with data and a pointer next which points to the structure node and create a pointer top and assign it with NULL
Step3 : stop

main()

Step1 : Start.
Step2 : Repeat the steps while condition is true.
Step3 : print " **** MENU **** "
Step4 : print " 1. Push 2. Pop 3. Display 4. Exit "
Step5 : print " Enter your choice : "
Step6 : Read choice.
Step7 : if choice = 1 print "Enter the value to be insert : " Read value. , push(value)
Step8 : if choice = 2 call the function pop()
Step9 : if choice = 3 call the function display()
Step10 : if choice = 4 call exit(0)
Step11 : Stop.

Push()

Step1 : Start
Step2 : set newnode = (struct node *) malloc(sizeof(struct node))
Step3 : newnode → data = value
Step4 : if top == NULL then newnode → next = NULL else. newnode → next = top
Step5 : top = newnode
Step6 : print " Insertion successfull "
Step7 : stop

pop().

Step1 : Start.
Step2 : If (top==NULL) print "Stack is empty!!!" else.
S' set struct node *temp= top
print "Deleted element:" temp→data
set top = temp→next.
free(temp)
Step3 : Stop

Display()

Step1 : Start.
Step3 : If top==NULL print "Stack is empty!!!" else.Step
Step4 : Set Struct node *temp = top.
Step5 : while temp→next != NULL Repeat the steps 6-
Step6 : print "temp→data "-->"
Step7 : Set temp = temp→next
Step8 : print temp→data "-->NULL"
Step9 : Stop.

Output.

**** MENU ****
1. PUSH
2. POP
3. Display
4. Exit
Enter your choice : 1
Enter the value to be Inserted : 25

Insertion is success.

**** MENU ****

1. Push
2. POP
3. Display
4. Exit.

Enter your choice : 1
Enter the value to be inserted : 10.

Insertion success !!!

**** MENU ****

1. Push
2. POP
3. Display
4. Exit

Enter your choice : 3.
10 --> 25 --> NULL

**** MENU ****

1. Push
2. POP
3. Display
4. Exit

Enter your choice : 2.

Deleted element : 10.

**** MENU ****

1. PUSH
2. POP.

3. Display
4. Exit.
Enter your choice : 3.
25 --> NULL

postfix expression

Algorithm.

main()

Step1 : Start
Step2 : Let i=0
Step3 : Print "Read the postfix expression :"
Step4 : Read pofx
Step5 : Repeat the steps 6-12. while ((ch=pofx[i++])!='\0
Step6 : if (isdigit(ch)) then push(ch-'0') else goto Step7
Step7 : op2=pop().
Step8 : set op1=pop()
Step9 : if ch='+' then push(op1+op2)
Step10 : if ch='-' then push(op1-op2)
Step11 : if ch='*' then push(op1*op2)
Step12 : if ch='/' then push(op1/op2)
Step13 : print "Given postfix expression is :", pofx
Step14 : print "Result after evaluation :", S[top]
Step15 : stop

push(int elem)

Step1 : Start
Step2 : Let S[++top]=elem
Step3 : Stop

pop()

Step1 : Start
Step3 : return S[top--]
Step4 : Stop.

output

Read the postfix expression:
53+62/*35*+

Given postfix expression is: 53+62/*35*+
Result after evaluation : 39.

queue using array

Algorithm.

main()

Step1 : Start
Step2 : Print "Enter the Size of QUEUE :"
Step3 : Read maxsize.
Step4 : print "Queue operations using array"
Step5 : print "1. Insert 2. Delete 3. Display 4. Exit"
Step6 : Repeat the steps 7-12. while choice <>4
Step7 : print "Enter your choice :"
Step8 : Read choice
Step9 : If choice=1 call function Insert()
Step10 : If choice=2 call the function dequeue()
Step11 : If choice=3 call the function display()
Step12 : If choice=4 exit()
Step13 : Stop.

Insert()

Step1 : Start
Step2 : print "Enter the element :".
Step3 : Read item.
Step4 : If (rear == maxsize-1) print "OVERFLOW"
Step5 : If front==-1 AND rear==-1 then Let front=0, rear=0
else goto Step 6
Step6 : rear = rear+1
Step7 : queue[rear] = item
Step8 : print "Value Inserted"
Step9 : Stop

dequeue()

Step1 : Start
Step2 : If front == -1 OR front > rear then print "UNDERFLOW"
else.
item = queue[front]
If (front == rear) then set front = -1, rear = -1 else
front = front + 1
print "value deleted"
Step3 : Stop.

display()

Step1 : Start
Step2 : If (rear == -1) print "Empty queue" else goto
step 3
Step3 : print "Elements in the queue are")
Step4 : Set i = front
Step5 : Repeat the steps 6-7 until i <= rear.
Step6 : print queue[i]
Step7 : Set i = i+1
Step8 : Stop.

Output

Queue operations using array.
1. Insert
2. Delete
3. Display
4. Exit.

Enter your choice : 1

Enter the element
6

Value Inserted
Enter your choice : 1

Enter the element
5

Value Inserted

Enter the element
4

Value Inserted.
Enter your choice : 3.

Elements in the queue are

6
5
4
Enter your choice : 2.
value deleted.
Enter your choice : 3.

Elements in the queue are

5
4