Charotar University of Science & Technology (CHARUSAT)

Devang Patel Institute of Advance Technology & Research (DEPSTAR)

CE245 Data Structure and Algorithms
Practical File

Semester:4th **Academic Year: 2022-23**

Practical- 1

Aim: Implement Linear Search and Binary Search using array data structure.

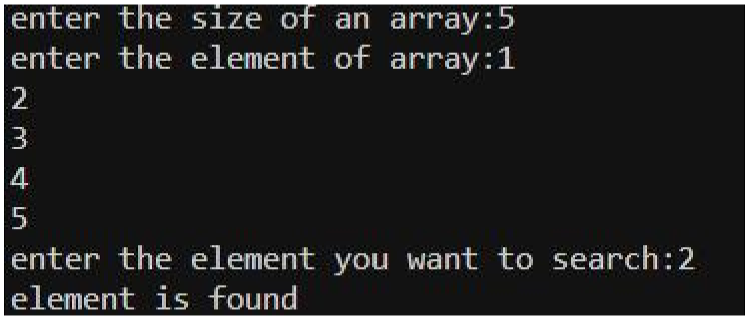
Solution :

Linear search:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int n;
   printf("enter the size of an array:");
   scanf("%d",&n);
   int a[n];
   printf("enter the element of array:");
   int i=0;
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   int b;
   int c;
   printf("enter the element you want to search:");
   scanf("%d",&b);
   for(i=0;i<n;i++)
   {
      if(a[i]==b)
      {
        c++;
      }

   }
```

```
    if(c>0)
    {
        printf("element is found \n");
    }
    else
    {
        printf("element is not found \n");
    }

}
```

Output:

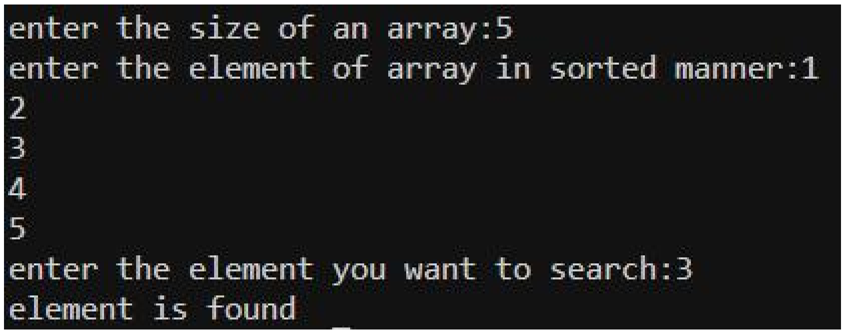


```
enter the size of an array:5
enter the element of array:1
2
3
4
5
enter the element you want to search:2
element is found
```

Binary search:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n;
    printf("enter the size of an array:");
    scanf("%d",&n);
    int a[n];
    printf("enter the element of array in sorted manner:");
    int i=0;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    int x;
    printf("enter the element you want to search:");
    scanf("%d",&x);

    int y=bin(a,n,x);
    if(y==-1)
    {
        printf("element is not found \n");
    }
    else
    {
        printf("element is found \n");
    }
}
int bin(int a[],int n,int x)
{
    int l=1,h=n;
    while(l<=h)
    {

        int mid=(l+h)/2;
        if(x==a[mid])
        {
            return mid;
        }
        else if(a[mid]>x)
        {
                h=mid-1;
        }
        else
        {
            l=mid+1;
        }

}
return -1;
}
```

Output:



```
enter the size of an array:5
enter the element of array in sorted manner:1
2
3
4
5
enter the element you want to search:3
element is found
```

Conclusion:

Practical- 4

Aim: Implement sorting Algorithm:

(a)bubble sort

(b)selection sort

(c)insertion sort

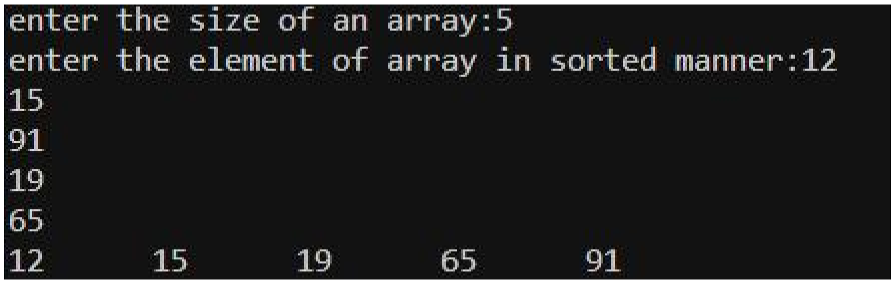
(d)quick sort

Solution:

(a) Bubble sort

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int n;
   printf("enter the size of an array:");
   scanf("%d",&n);
   int a[n];
   printf("enter the element of array in sorted manner:");
   int i=0;
   int j=0;
   for(i=0;i<n;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])
         {
            int temp;
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
         }
      }
   }
   for(i=0;i<n;i++)
   {
      printf("%d \t",a[i]);
   }
}
```

Output:

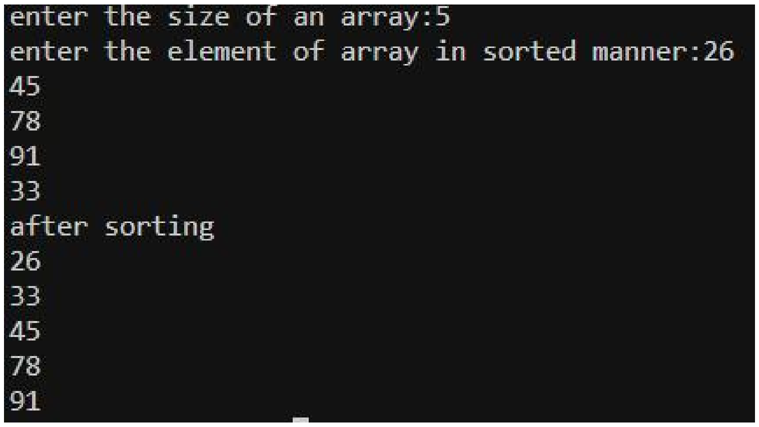


```
enter the size of an array:5
enter the element of array in sorted manner:12
15
91
19
65
12      15      19      65      91
```

(b) Selection sort:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n;
    printf("enter the size of an array:");
    scanf("%d",&n);
    int a[n];
    printf("enter the element of array in sorted manner:");
    int i=0;
    int j=0;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<n-1;i++)
    {
        int min=i;
        for(j=i+1;j<n;j++)
        {
            if(a[j]<a[min])
            {
                min=j;
            }
        }
        if(min!=i)
            {
                int temp;
                temp=a[i];
                a[i]=a[min];
                a[min]=temp;
            }
    }
    printf("after sorting \n");
    for(i=0;i<n;i++)
    {
       printf("%d \n",a[i]);
    }
}
```

Output:

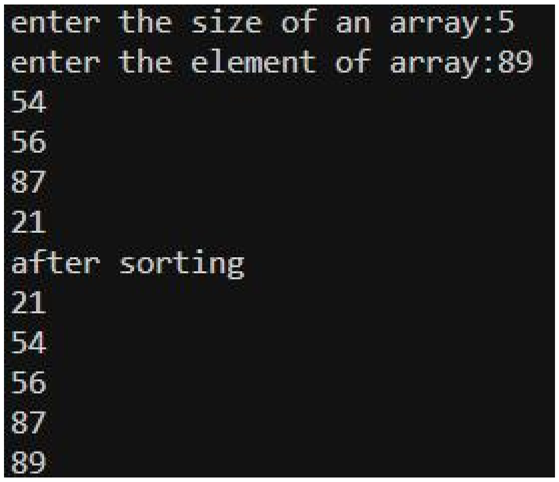


```
enter the size of an array:5
enter the element of array in sorted manner:26
45
78
91
33
after sorting
26
33
45
78
91
```

(c) Insertion sort:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n;
    printf("enter the size of an array:");
    scanf("%d",&n);
    int a[n];
    printf("enter the element of array:");
    int i=0;
    int j=0;
    int temp;
    for(i=0;i<n;i++)
    {
       scanf("%d",&a[i]);
    }
    for(i=1;i<n;i++)
    {
        temp=a[i];
       j=i-1;
       while(j>=0 && a[j]>temp)
       {
          a[j+1]=a[j];
          j--;
       }
       a[j+1]=temp;
    }
    printf("after sorting \n");
    for(i=0;i<n;i++)
    {
        printf("%d \n",a[i]);
    }}
```

Output:

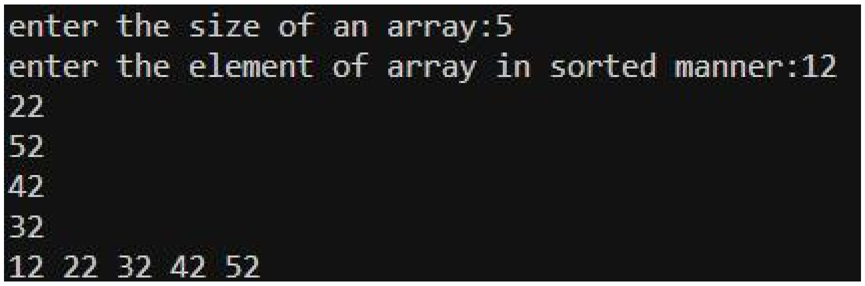


```
enter the size of an array:5
enter the element of array:89
54
56
87
21
after sorting
21
54
56
87
89
```

(d) Quick sort:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n;
    printf("enter the size of an array:");
    scanf("%d",&n);
    int a[n];
    printf("enter the element of array in sorted manner:");
    int i=0;
    int j=0;
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    quick(a,0,n);
    for(i=0;i<n;i++)
    {
        printf("%d \n ",a[i]);
    }
}
void quick(int a[],int l,int h)
{
    if(l<h)
    {
        int loc=part(a,l,h);
        quick(a,l,loc-1);
        quick(a,loc+1,h);
    }
}
int part(int a[],int l,int h)
{
   int pivot=a[l];
   int i=l;
   int j=h;
   int temp;
   while(i<j)
   {
      while(a[i]<=pivot)
      {
         i++;
      }
      while(a[j]>pivot)
      {
         j--;
      }
      if(i<j)
      {
         temp=a[i];
         a[i]=a[j];
         a[j]=temp;
      }
   }
   temp=a[l];
   a[l]=a[j];
   a[j]=temp;
   return j;
}
```

Output:



```
enter the size of an array:5
enter the element of array in sorted manner:12
22
52
42
32
12 22 32 42 52
```

Conclusion:

Practical- 9

Aim: Chef has a sequence A1, A2, ... AN and an integer K. Now there is a sliding window of size K which is moving from the very left of the array to the very right and at a particular time Chef has access to only those elements that are present in that window and Chef wants to find the number of the distinct elements of each window of size K. Help Chef to find the answer.

Input:

The first line of the input contains a single integer T denoting the number of test cases.

The description of T test cases follows.

The first line of each test case contains two integers N and K.

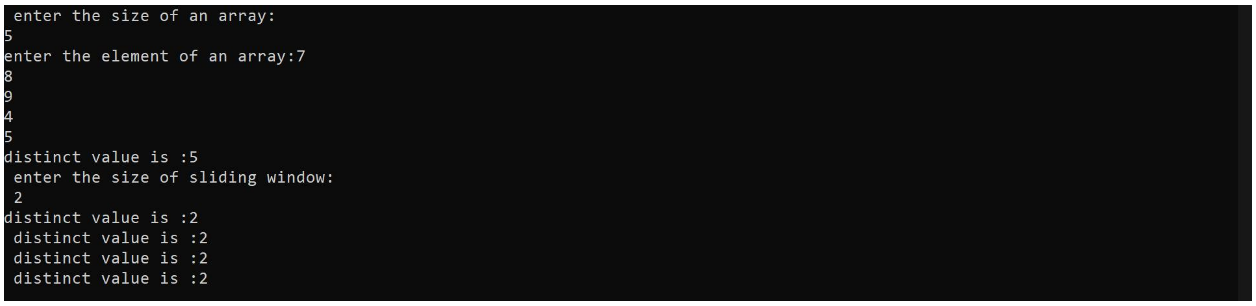
The second line contains N space-separated integers A1,A2,A3,.....AN Output

For each test case, print a single line containing space-separated integers — the number of the distinct elements of each window of size from the very left of the array to the very right of the sequence.

Solution :

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int s;
   printf("enter the value of test case:");
   scanf("%d",&s);
   int u=0;
   while(u<s)
   {
   printf("enter the size of an array:\n");
   int n;
   scanf("%d",&n);
   int a[n];
   printf("enter the element of an array:");
   int i=0;
   int j;
   int k;
   int c=n;
   for(i=0;i<n;i++)
   {
        scanf("%d",&a[i]);
  for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]==a[j])
            {
                c--;
                break;
            }
        }
    }
    printf("distinct value is :%d \n ",c);
    printf("enter the size of sliding window:\n ");
    int x;
    scanf("%d",&x);
    int y;
    for(k=0;k<n-x+1;k++)
    {
        y=x;
            for(i=k;i<x+k;i++)
            {
                for(j=i+1;j<x+k;j++)
                {
                    if(a[i]==a[j])
                    {
                        y--;
                        break;
                    }
                }
            }
            printf("distinct value is :%d \n ",y);
    }u++;
    }
    return 0;
}
```

Output:



```
 enter the size of an array:
5
enter the element of an array:7
8
9
4
5
distinct value is :5
 enter the size of sliding window:
 2
distinct value is :2
 distinct value is :2
 distinct value is :2
 distinct value is :2
```

Conclusion:

Practical-3

Aim:

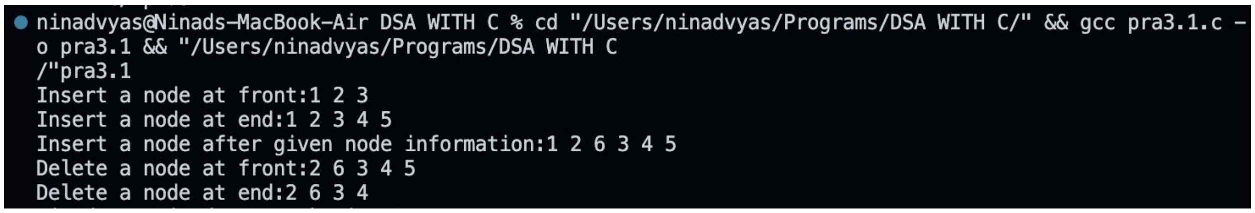
3.1 Implement following operations of singly linked list.
(a) Insert a node at front
(b) Insert a node at end
(c) Insert a node after given node information
(d) Delete a node at front
(e) Delete a node at last
3.2 Implement following operations of doubly linked list.
(a) Insert a node at front
(b) Insert a node at end
(c) Insert a node after given node information
(d) Delete a node at front
(e) Count number of nodes
3.3 Implement following operations of circular singly linked list.
(a) Inserting a node at front
(b) Delete a node at end

Solution(3.1):

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node* next;
};
void insertAtFront(struct node** headRef, int newData) {
struct node* newNode = (struct node*)malloc(sizeof(struct node));
newNode->data = newData;
newNode->next = (*headRef);
(*headRef) = newNode;
}
void insertAtEnd(struct node** headRef, int newData) {
struct node* newNode = (struct node*)malloc(sizeof(struct node));
newNode->data = newData;
newNode->next = NULL;
if (*headRef == NULL) {
*headRef = newNode;
return;
}
struct node* temp = *headRef;
while (temp->next != NULL) {
temp = temp->next;
}
temp->next = newNode;
}
```

```
void insertAfter(struct node* prevNode, int newData) {
if (prevNode == NULL) {
printf("Error: Previous node cannot be NULL");
return;
}
struct node* newNode = (struct node*)malloc(sizeof(struct node));
newNode->data = newData;
newNode->next = prevNode->next;
prevNode->next = newNode;
}
void deleteAtFront(struct node** headRef) {
if (*headRef == NULL) {
printf("Error: List is empty");
return;
}
struct node* temp = *headRef;
*headRef = (*headRef)->next;
free(temp);
}
void deleteAtEnd(struct node** headRef) {
if (*headRef == NULL) {
printf("Error: List is empty");
return;
}
if ((*headRef)->next == NULL) {
free(*headRef);
*headRef = NULL;
return;
}
struct node* temp = *headRef;
while (temp->next->next != NULL) {
temp = temp->next;
}
free(temp->next);
temp->next = NULL;
}
void printList(struct node* head) {
while (head != NULL) {
printf("%d ", head->data);
head = head->next;
}
printf("\n");
}
int main() {
struct node* head = NULL;
printf("Insert a node at front:");
insertAtFront(&head, 3);
insertAtFront(&head, 2);
insertAtFront(&head, 1);
printList(head); // Output: 1 2 3
printf("Insert a node at end:");
insertAtEnd(&head, 4);
insertAtEnd(&head, 5);
printList(head); // Output: 1 2 3 4 5
printf("Insert a node after given node information:");
insertAfter(head->next, 6);
printList(head); // Output: 1 2 6 3 4 5
printf("Delete a node at front:");
deleteAtFront(&head);
printList(head); // Output: 2 6 3 4 5
printf("Delete a node at end:");
deleteAtEnd(&head);
printList(head); // Output: 2 6 3 4
return 0;
}
```

Output:



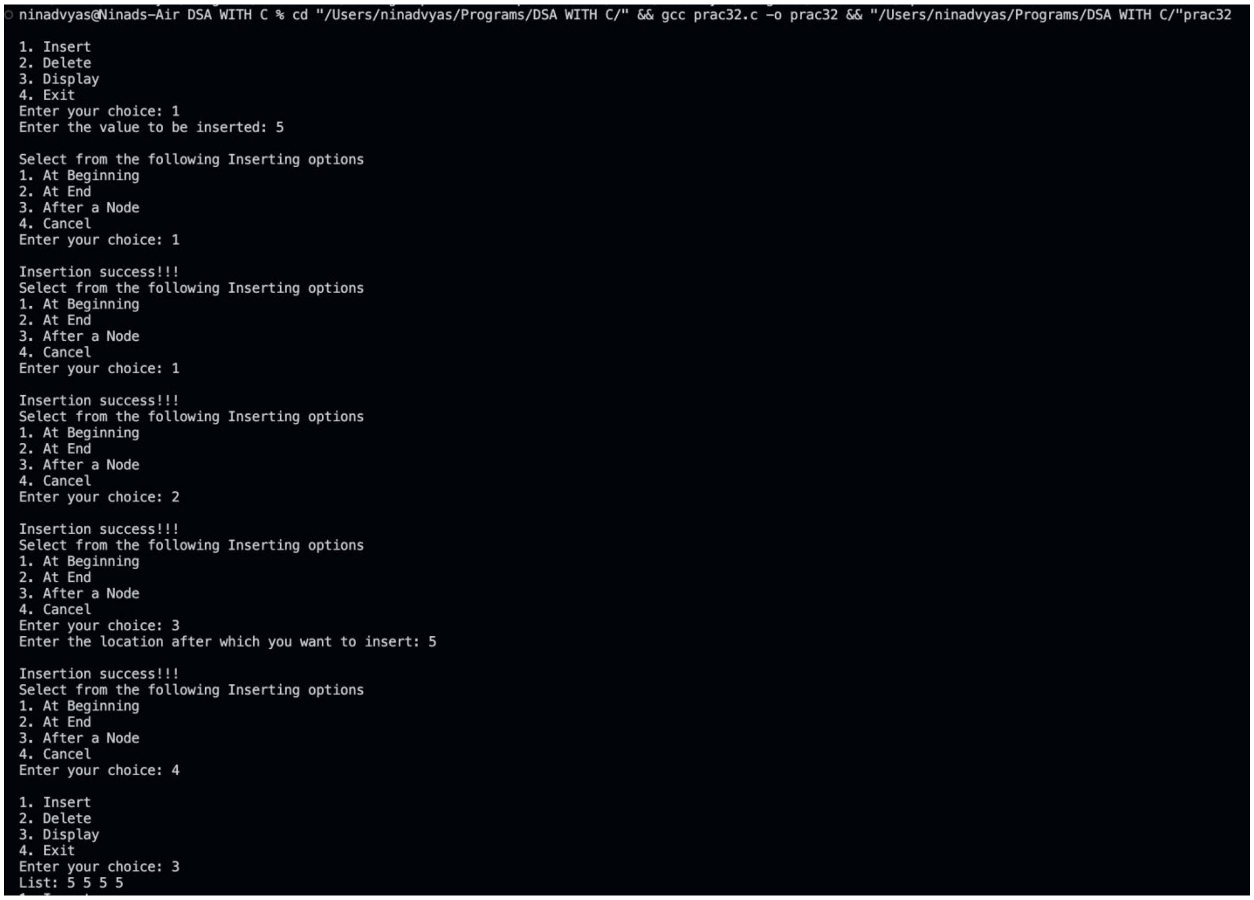
```
● ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc pra3.1.c -
  o pra3.1 && "/Users/ninadvyas/Programs/DSA WITH C
  /"pra3.1
  Insert a node at front:1 2 3
  Insert a node at end:1 2 3 4 5
  Insert a node after given node information:1 2 6 3 4 5
  Delete a node at front:2 6 3 4 5
  Delete a node at end:2 6 3 4
```

Solution(3.2):

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
int data;
struct Node* previous;
struct Node* next;
}*head = NULL;
void insertAtBeginning(int value)
{
struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
newNode->data = value;
newNode->previous = NULL;
if (head == NULL)
{
newNode->next = NULL;
}
else
{
head->previous = newNode;
newNode->next = head;
}
head = newNode;
printf("\nInsertion success!!!");
}
void insertAtEnd(int value)
{
struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
if (head == NULL)
{
newNode->previous = NULL;
head = newNode;
}
else
{
struct Node* temp = head;
while (temp->next != NULL)
temp = temp->next;
temp->next = newNode;
newNode->previous = temp;
}
printf("\nInsertion success!!!");
}
void insertAtAfter(int value, int location)
{
if (head == NULL)
{
printf("List is empty. Cannot insert after a node.");
return;
}
struct Node* temp = head;
while (temp != NULL && temp->data != location)
temp = temp->next;
if (temp == NULL)
{
printf("Node not found. Cannot insert after the node.");
return;
}
struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
newNode->data = value;
newNode->previous = temp;
newNode->next = temp->next;
if (temp->next != NULL)
temp->next->previous = newNode;
temp->next = newNode;
printf("\nInsertion success!!!");
```

```
}
void deleteBeginning()
{
if (head == NULL)
{
printf("List is empty. Cannot delete the first node.");
return;
}
struct Node* temp = head;
head = head->next;
if (head != NULL)
head->previous = NULL;
free(temp);
printf("\nDeletion success!!!");
}
int countNodes()
{
int count = 0;
struct Node* temp = head;
while (temp != NULL)
{
count++;
temp = temp->next;
}
return count;
}
void display()
{
if (head == NULL)
{
printf("List is empty. Nothing to display.");
return;
}
struct Node* temp = head;
printf("List: ");
while (temp != NULL)
{
printf("%d ", temp->data);
temp = temp->next;
} }
int main()
{
int choice1, choice2, value, location;
while(1)
{
printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");
scanf("%d",&choice1);
switch(choice1)
{
case 1: printf("Enter the value to be inserted: ");
scanf("%d",&value);
while(1)
{
printf("\nSelect from the following Inserting options\n");
printf("1. At Beginning\n2. At End\n3. After a Node\n4. Cancel\nEnter your choice: ");
scanf("%d",&choice2);
switch(choice2)
{
case 1: insertAtBeginning(value);
break;
case 2: insertAtEnd(value);
break;
case 3: printf("Enter the location after which you want to insert: ");
scanf("%d",&location);
insertAtAfter(value,location);
break;
case 4: goto EndSwitch;
default: printf("\nPlease select correct Inserting option!!!\n");
}
}
case 2: while(1)
{
printf("\nSelect from the following Deleting options\n");
printf("1. At Beginning\n2. At End\n3. Specific Node\n4. Cancel\nEnter your choice: ");
scanf("%d",&choice2);
switch(choice2)
{
case 1: deleteBeginning();
break;
case 2:
break;
case 3:
break;
case 4: goto EndSwitch;
default: printf("\nPlease select correct Deleting option!!!\n");
}
}
EndSwitch: break;
case 3: display();
break;
case 4: exit(0);
default: printf("\nPlease select correct option!!!");
}
}
return 0;
}
```

Output:



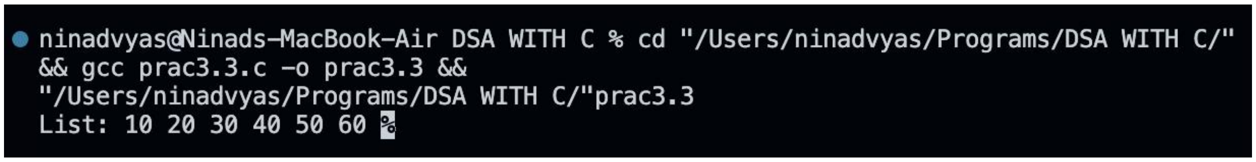
```
○ ninadvyas@Ninads-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc prac32.c -o prac32 && "/Users/ninadvyas/Programs/DSA WITH C/"prac32

 1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter your choice: 1
 Enter the value to be inserted: 5

 Select from the following Inserting options
 1. At Beginning
 2. At End
 3. After a Node
 4. Cancel
 Enter your choice: 1

 Insertion success!!!
 Select from the following Inserting options
 1. At Beginning
 2. At End
 3. After a Node
 4. Cancel
 Enter your choice: 1

 Insertion success!!!
 Select from the following Inserting options
 1. At Beginning
 2. At End
 3. After a Node
 4. Cancel
 Enter your choice: 2

 Insertion success!!!
 Select from the following Inserting options
 1. At Beginning
 2. At End
 3. After a Node
 4. Cancel
 Enter your choice: 3
 Enter the location after which you want to insert: 5

 Insertion success!!!
 Select from the following Inserting options
 1. At Beginning
 2. At End
 3. After a Node
 4. Cancel
 Enter your choice: 4

 1. Insert
 2. Delete
 3. Display
 4. Exit
 Enter your choice: 3
 List: 5 5 5 5
```

Solution[3.3(a)]:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node* createNode(){
struct node *n;
n = (struct node*)malloc(sizeof(struct node));
return n;
}

void insertNodeAtBeginning(struct node** head, int data){
struct node* temp, *current;
temp = createNode();
temp->data = data;
temp->next = NULL;
if(*head == NULL){
```

```
*head = temp;
temp->next = *head;
return;
}
current = *head;
while(current->next != *head){
current = current->next;
}
current->next = temp;
temp->next = *head;
*head = temp;
return;
}
void viewList(struct node** head){
struct node *temp;
if(*head == NULL){
printf("List is empty.\n");
return;
}else{
temp = *head;
printf("List: ");
do{
printf("%d ", temp->data);
temp = temp->next;
}
while(temp != *head);
}
}
int main(){
struct node* head = NULL;
insertNodeAtBeginning(&head, 60);
insertNodeAtBeginning(&head, 50);
insertNodeAtBeginning(&head, 40);
insertNodeAtBeginning(&head, 30);
insertNodeAtBeginning(&head, 20);
insertNodeAtBeginning(&head, 10);
viewList(&head);
return 0;
}
```

Output:

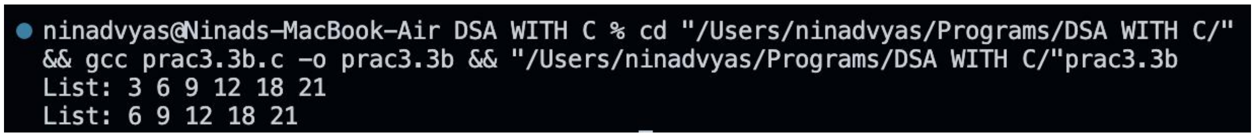


```
ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/"
&& gcc prac3.3.c -o prac3.3 &&
"/Users/ninadvyas/Programs/DSA WITH C/"prac3.3
List: 10 20 30 40 50 60 %
```

Solution[3.3(b)]:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node* createNode(){
struct node *n;
n = (struct node*)malloc(sizeof(struct node));
return n;
}
void insertNodeAtEnd(struct node** head, int data){
struct node* temp, *current;
temp = createNode();
temp->data = data;
temp->next = NULL;
if(*head == NULL){
*head = temp;
temp->next = *head;
return;
}
current = *head;
while(current->next != *head){
current = current->next;
}
current->next = temp;
temp->next = *head;
return;
}
void deleteFirstNode(struct node** head){
struct node* temp = *head;
struct node* current = *head;
if(*head == NULL){
printf("List empty.");
return;
}else if((*head)->next == *head){
*head = NULL;
free(temp);
return;
}
while(current->next != *head){
current = current->next;
}
current->next = (*head)->next;
*head = (*head)->next;
free(temp);
return;
}
void viewList(struct node** head){
struct node *temp;
if(*head == NULL){
printf("List is empty.\n");
return;
}else{ temp = *head;
printf("List: ");
do{
printf("%d ", temp->data);
temp = temp->next;
}
while(temp != *head);
}
printf("\n");
}
int main(){
struct node* head = NULL;
insertNodeAtEnd(&head, 3);
insertNodeAtEnd(&head, 6);
insertNodeAtEnd(&head, 9);
insertNodeAtEnd(&head, 12);
insertNodeAtEnd(&head, 18);
insertNodeAtEnd(&head, 21);
viewList(&head);
deleteFirstNode(&head);
viewList(&head);
return 0;
}
```

Output:



```
ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/"
&& gcc prac3.3b.c -o prac3.3b && "/Users/ninadvyas/Programs/DSA WITH C/"prac3.3b
List: 3 6 9 12 18 21
List: 6 9 12 18 21
```

Conclusion:

Practical-6

Aim:

6.1 Implement basic operations (push (), pop () and display ()) of stack using array.

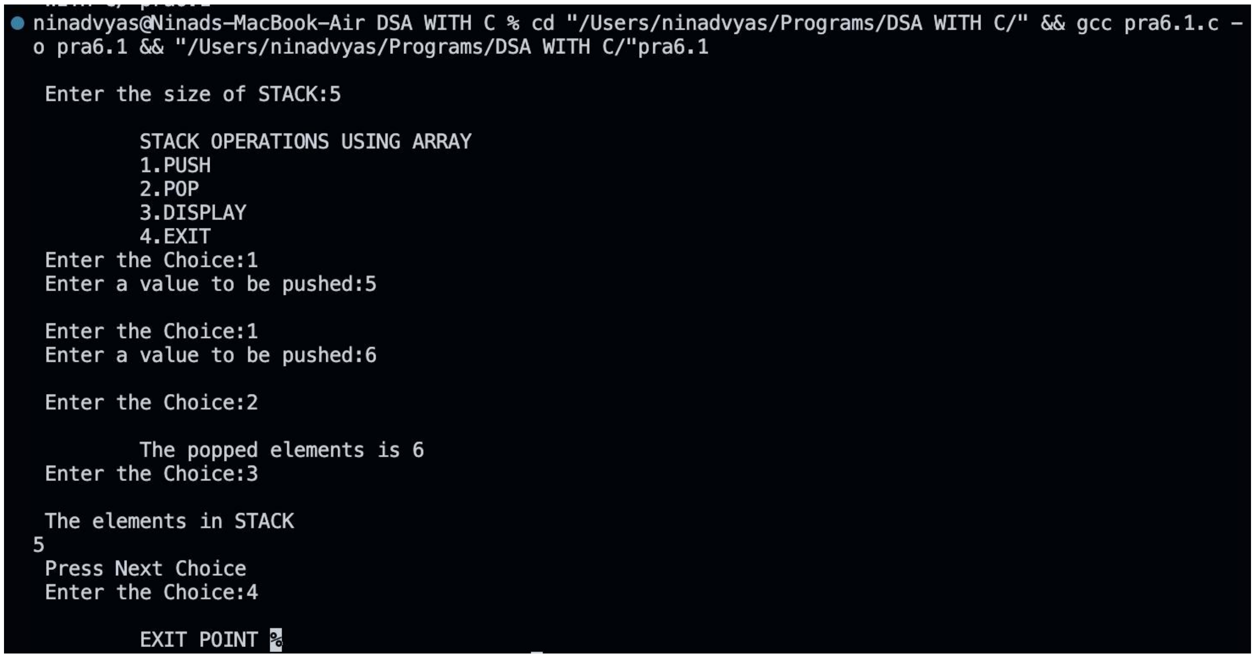
6.2 Implement basic operations (push (), pop () and display ()) of stack using linked list.

Solution(6.1):

```
#include<stdio.h>
int stack[5],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
top=-1;
printf("\n Enter the size of STACK:");
scanf("%d",&n);
printf("\n\t STACK OPERATIONS USING ARRAY");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
do
{
printf("\n Enter the Choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
printf("\n\t EXIT POINT ");
break;
}
default:
```

```
{
printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
}}}
while(choice!=4);
return 0;
}
void push()
{
if(top>=n-1)
{
printf("\n\tSTACK is over flow");
}
else
{
printf(" Enter a value to be pushed:");
scanf("%d",&x);
top++;
stack[top]=x;
}
}
void pop()
{
if(top<=-1)
{
printf("\n\t Stack is under flow");
}
else
{
printf("\n\t The popped elements is %d",stack[top]);
top--;
}
}
void display()
{
if(top>=0)
{
printf("\n The elements in STACK ");
for(i=top; i>=0; i--)
printf("\n%d",stack[i]);
printf("\n Press Next Choice");
}
else
{
printf("\n The STACK is empty");
}}
```

Output:



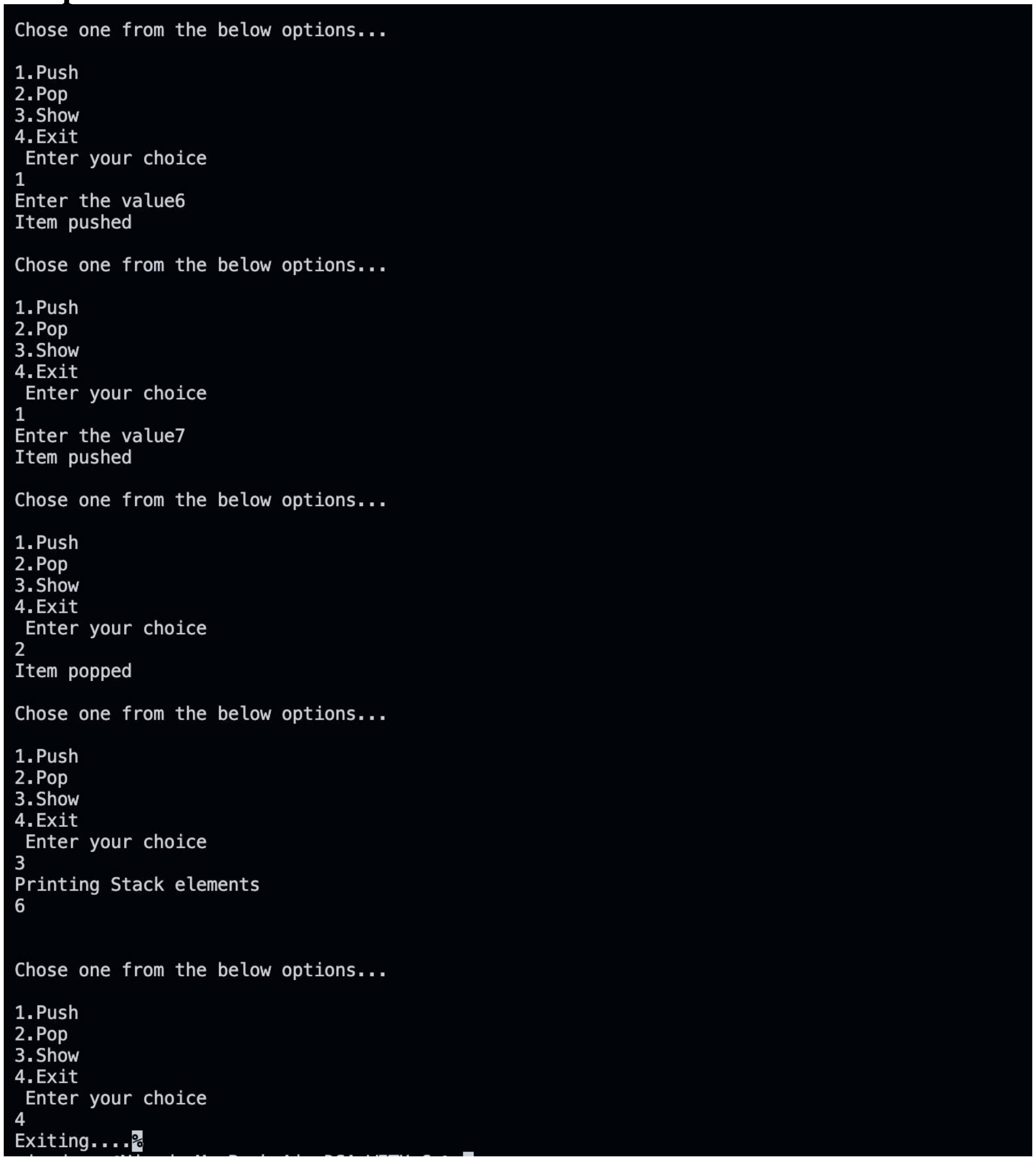
```
● ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc pra6.1.c -
  o pra6.1 && "/Users/ninadvyas/Programs/DSA WITH C/"pra6.1

   Enter the size of STACK:5

           STACK OPERATIONS USING ARRAY
           1.PUSH
           2.POP
           3.DISPLAY
           4.EXIT
   Enter the Choice:1
   Enter a value to be pushed:5

   Enter the Choice:1
   Enter a value to be pushed:6

   Enter the Choice:2

           The popped elements is 6
   Enter the Choice:3

   The elements in STACK
  5
   Press Next Choice
   Enter the Choice:4

           EXIT POINT %
```

Solution(6.2):

```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

void main ()
{
int choice=0;
while(choice != 4)
{
printf("\n\nChose one from the below options...\n");
printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
printf("\n Enter your choice \n");
scanf("%d",&choice);
switch(choice)
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
printf("Exiting....");
break;
}
default:
{
printf("Please Enter valid choice ");
}
};
}
}
void push ()
{
int val;
struct node *ptr = (struct node*)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("not able to push the element");
}
else
{
printf("Enter the value");
scanf("%d",&val);
if(head==NULL)
{
ptr->val = val;
ptr -> next = NULL;
head=ptr;
}
else
{
ptr->val = val;
ptr->next = head;
head=ptr;
}
```

```
printf("Item pushed");
}
}
void pop()
{
int item;
struct node *ptr;
if (head == NULL)
{
printf("Underflow");
}
else
{
item = head->val;
ptr = head;
head = head->next;
free(ptr);
printf("Item popped");
}
}
void display()
{
int i;
struct node *ptr;
ptr=head;
if(ptr == NULL)
{
printf("Stack is empty\n");
}
else
{
printf("Printing Stack elements \n");
while(ptr!=NULL)
{
printf("%d\n",ptr->val);
ptr = ptr->next;
}
}
}
```

Output:



```
Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
1
Enter the value6
Item pushed

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
1
Enter the value7
Item pushed

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
2
Item popped

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
3
Printing Stack elements
6


Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
4
Exiting....%
```

Conclusion:

Practical-8

Aim:

8.1 Implement basic operations (enqueue (), dequeue () and display ()) of queue using array.

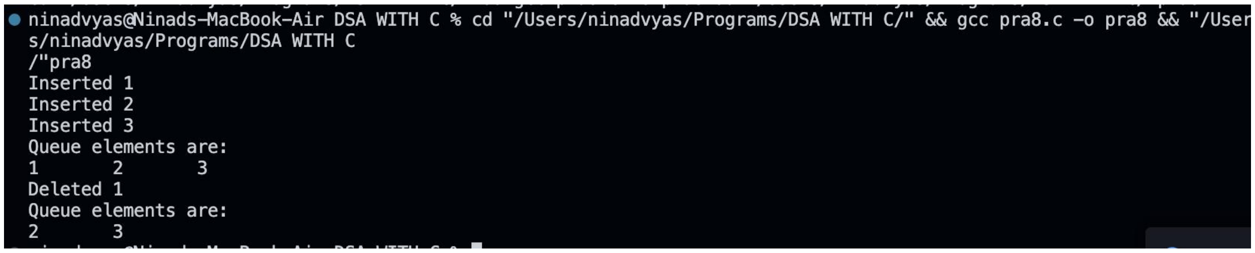
8.2 Implement basic operations (enqueue (), dequeue () and display ()) of queue using linked list.

8.3 Implement basic operations (enqueue (), dequeue () and display ()) of circular queue using array.

Solution(8.1):

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 10

struct Queue {
int items[MAX_SIZE];
int front;
int rear;
};
typedef struct Queue queue;
void enqueue(queue *q, int item) {
if (q->rear == MAX_SIZE - 1) {
printf("Queue is full\n");
} else {
if (q->front == -1) {
q->front = 0;
}
q->rear++;
q->items[q->rear] = item;
printf("Inserted %d\n", item);
}
}
int dequeue(queue *q) {
int item;
if (q->front == -1 || q->front > q->rear) {
printf("Queue is empty\n");
return -1;
} else {
item = q->items[q->front];
q->front++;
printf("Deleted %d\n", item);
return item;
}
}
void display(queue *q) {
int i;
if (q->rear == -1) {
printf("Queue is empty\n");
} else {
printf("Queue elements are:\n");
for (i = q->front; i <= q->rear; i++) {
printf("%d\t", q->items[i]);
}
printf("\n");
}
}
int main() {
queue *q = (queue *)malloc(sizeof(queue));
q->front = -1;
q->rear = -1;
enqueue(q, 1);
enqueue(q, 2);
enqueue(q, 3);
display(q);
dequeue(q);
display(q);
return 0;
}
```

Output:



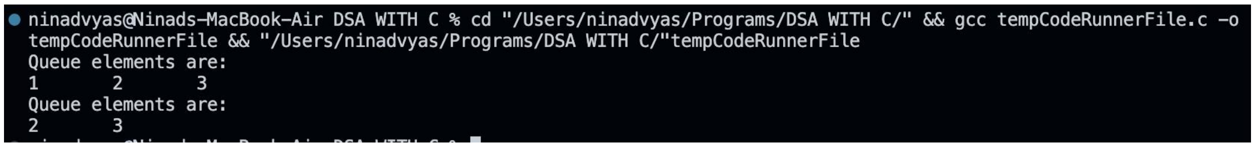
```
ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc pra8.c -o pra8 && "/User
s/ninadvyas/Programs/DSA WITH C
/"pra8
Inserted 1
Inserted 2
Inserted 3
Queue elements are:
1       2       3
Deleted 1
Queue elements are:
2       3
```

Solution(8.2):

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
int data;
struct Node* next;
};

struct Queue {
struct Node *front, *rear;
};
```

```
typedef struct Queue queue;

void enqueue(queue *q, int item) {
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = item;
newNode->next = NULL;
if (q->rear == NULL) {
q->front = q->rear = newNode;
return;
}
q->rear->next = newNode;
q->rear = newNode;
}

int dequeue(queue *q) {
if (q->front == NULL) {
printf("Queue is empty\n");
return -1;
}
struct Node* temp = q->front;
int item = temp->data;
q->front = q->front->next;
if (q->front == NULL) {
q->rear = NULL;
}
free(temp);
return item;
}

void display(queue *q) {
struct Node* temp = q->front;
if (q->front == NULL) {
printf("Queue is empty\n");
return;
}
printf("Queue elements are:\n");
while (temp != NULL) {
printf("%d\t", temp->data);
temp = temp->next;
}
printf("\n");
}

int main() {
queue *q = (queue *)malloc(sizeof(queue));
q->front = q->rear = NULL;
enqueue(q, 1);
enqueue(q, 2);
enqueue(q, 3);
display(q);
dequeue(q);
display(q);
return 0;
}
```

Output:

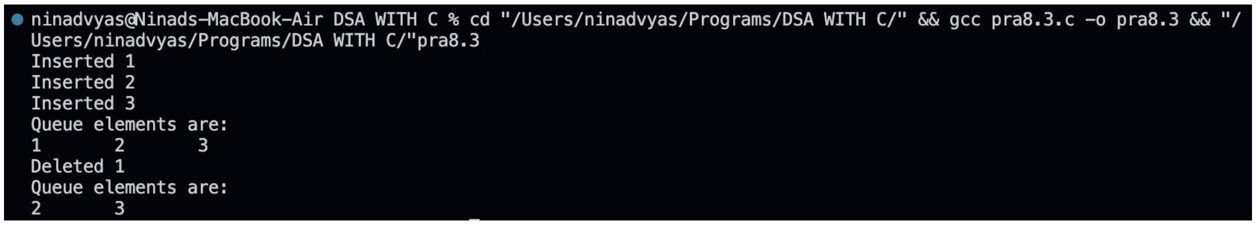


```
ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc tempCodeRunnerFile.c -o
tempCodeRunnerFile && "/Users/ninadvyas/Programs/DSA WITH C/"tempCodeRunnerFile
Queue elements are:
1       2       3
Queue elements are:
2       3
```

Solution(8.3):

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 10
struct CircularQueue {
int items[MAX_SIZE];
int front, rear;
};
typedef struct CircularQueue circularQueue;
void enqueue(circularQueue *q, int item) {
if ((q->rear + 1) % MAX_SIZE == q->front) {
printf("Queue is full\n");
} else {
if (q->front == -1) {
q->front = 0;
}
q->rear = (q->rear + 1) % MAX_SIZE;
q->items[q->rear] = item;
printf("Inserted %d\n", item);
}}
int dequeue(circularQueue *q) {
int item;
if (q->front == -1) {
printf("Queue is empty\n");
return -1;
} else {
item = q->items[q->front];
if (q->front == q->rear) {
q->front = q->rear = -1;
} else {
q->front = (q->front + 1) % MAX_SIZE;
}
printf("Deleted %d\n", item);
return item;
}}
void display(circularQueue *q) {
int i;
if (q->rear == -1) {
printf("Queue is empty\n");
} else {
printf("Queue elements are:\n");
for (i = q->front; i <= q->rear; i = (i + 1) % MAX_SIZE) {
printf("%d\t", q->items[i]);
}
printf("\n");
}}
int main() {
circularQueue *q = (circularQueue *)malloc(sizeof(circularQueue));
q->front = q->rear = -1;
enqueue(q, 1);
enqueue(q, 2);
enqueue(q, 3);
display(q);
dequeue(q);
display(q);
return 0;
}
```

Output:



```
● ninadvyas@Ninads-MacBook-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc pra8.3.c -o pra8.3 && "/
  Users/ninadvyas/Programs/DSA WITH C/"pra8.3
  Inserted 1
  Inserted 2
  Inserted 3
  Queue elements are:
  1       2       3
  Deleted 1
  Queue elements are:
  2       3
```

Conclusion:

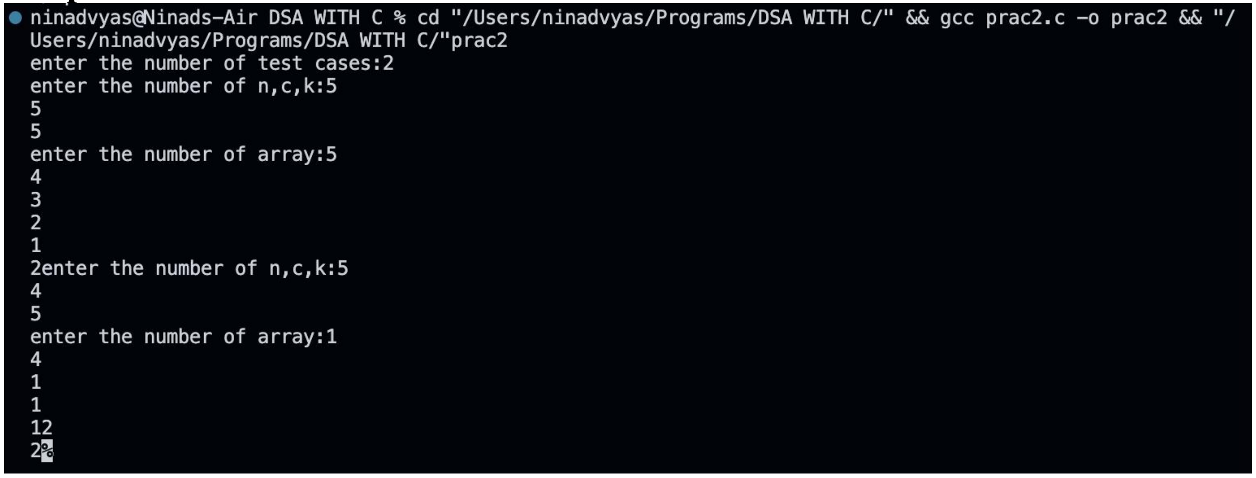
Practical- 2

In a far away Galaxy of Tilky Way, there was a planet Tarth where the sport of Competitive Coding was very popular. According to legends, there lived a setter known for loving knapsack type problems.
Given N objects in a row, with weights W1,W2,...,WN, you need to find the maximum number of consecutive objects you can fill in a bag of maximum capacity C such that the total weight of objects taken is at least K.
In other words, pick objects such that-The total weight of collected objects is at least K.
The total weight does not exceed C.
The objects picked must be consecutive (i.e. a subarray of the objects need to be picked) The number of objects is maximized. You need to print this maximum value.
Note-If no such object could be picked, then the answer is obviously 0.
Input
The first line of input contains T, number of test cases in a file.
The next line contains three integers, N, C and K, as described in the problem statement.
The next line contains N space separated integers, denoting Wi, i.e. weight of the object.
Output
For test case, output the maximum number of objects you can pick.

Solution :
```
#include<stdio.h>
int main() {
    int t;
    int  value[100];
    int n,k,c;
    printf("enter the number of test cases:");
    scanf("%d",&t);
     int sum=0,count=0,max=0;
     for(int i=0;i<t;i++){
      printf("enter the number of n,c,k:");
      scanf("%d %d %d",&n,&k,&c);
      printf("enter the number of array:");
      for(int i=0;i<n;i++)
```

```
        {
          scanf("%d",&value[i]);
        }

        for(int i=0;i<n;i++)
        {
          for(int j=i;j<n;j++)
          {
            sum=sum+value[j];
            count++;
            if(sum>=k && sum<=c && count>max)
{
                max=count;
            }
            else if(sum>c){
                break;
            }
          }
        sum=0;
        count=0;
      }
      printf("%d",max);
 }
   return 0;
}
```

Output:



```
● ninadvyas@Ninads-Air DSA WITH C % cd "/Users/ninadvyas/Programs/DSA WITH C/" && gcc prac2.c -o prac2 && "/
  Users/ninadvyas/Programs/DSA WITH C/"prac2
  enter the number of test cases:2
  enter the number of n,c,k:5
  5
  5
  enter the number of array:5
  4
  3
  2
  1
  2enter the number of n,c,k:5
  4
  5
  enter the number of array:1
  4
  1
  1
  12
  2%
```

Conclusion: