# Practical 1

1. **Title: -** Write a program to print 5 numbers entered by user in an array.
2. **Outcome: -** print the 5 elements entered by user.
3. **Objectives: -** Understand the concept of Array and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

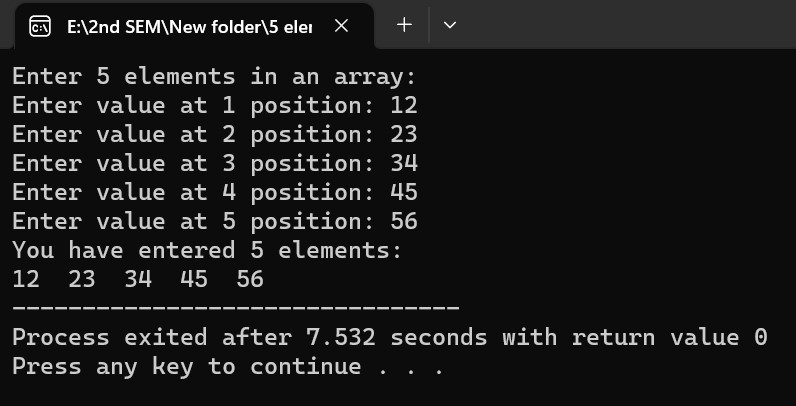
|  |  |
| --- | --- |
| a | Name of an array |
| i | Counter for loop |

## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include<iostream> using namespace std; int main()  {  int i,a[5];  cout<<"Enter 5 elements in an array: "<<endl;  for(i=0;i<5;i++)  {  cout<<"Enter value at "<<i+1<<" position: ";  cin>>a[i];  }  cout<<"You have entered 5 elements: "<<endl; for(i=0;i<5;i++)  {  cout<<a[i]<<" ";  }  return 0;    } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: Understanding how to enter the elements in an array. 4.3.2.2 Issues: N.A

# Practical 2

1. **Title: -** Write a program to print sum of 5 elements of an array entered by user.
2. **Outcome: -** Sum of an array elements.
3. **Objectives: -** Understand the concept of Arrayand variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

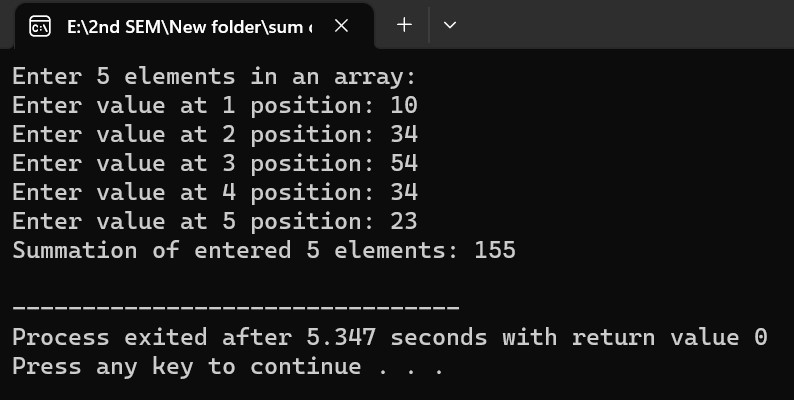
|  |  |
| --- | --- |
| a | Name of an array |
| i | Counter for loop |
| Sum | Sum of elements |

## 4.2 Code/ Pseudo Code

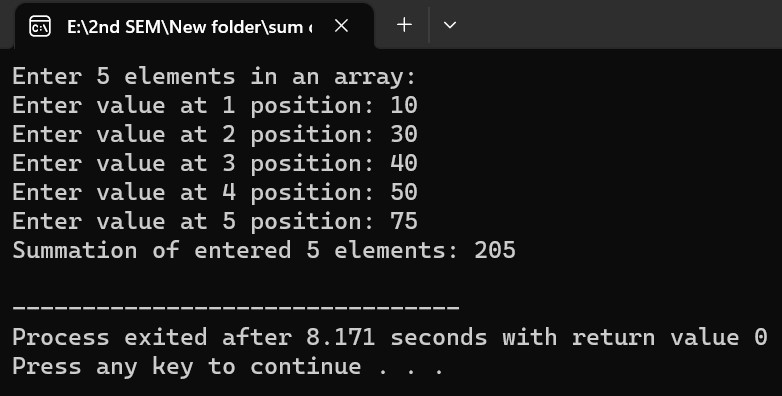
|  |
| --- |
| #include<iostream> using namespace std;  int main()  {  int i,a[5],sum=0;  cout<<"Enter 5 elements in an array: "<<endl;  for(i=0;i<5;i++)  {  cout<<"Enter value at "<<i+1<<" position: ";  cin>>a[i];  }  for(i=0;i<5;i++)  {  sum=sum+a[i];  }  cout<<"Summation of entered 5 elements: "<<sum<<endl; return 0;  } |

## 4.3 Results

**4.3.1 Test Case 1:**



**4.3.2 Test case 2:**



### 4.3.3 Result Analysis

4.3.3.1 Advantages: printing sum of an array.

4.3.3.2 Issues: N.A

# Practical 3

1. **Title: -** Write a program to print average of 5 elements of an array entered by user.
2. **Outcome: -** Average of an array elements.
3. **Objectives: -** Understand the concept of Array and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

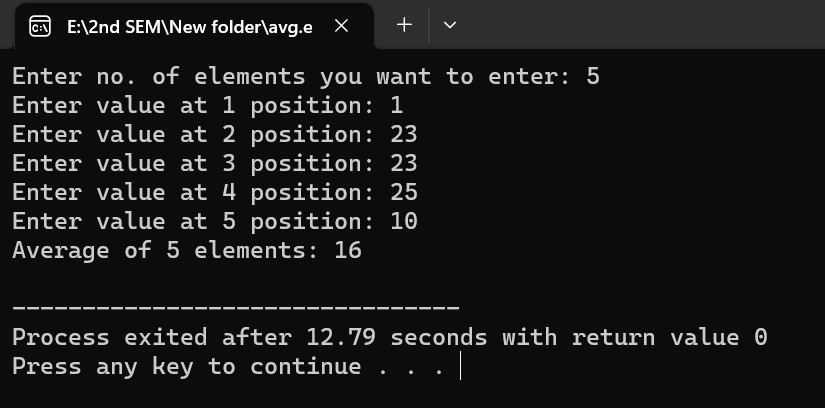
|  |  |
| --- | --- |
| a | Name of an array |
| i | Counter for loop |
| n | No. of elements |
| sum | Sum of elements |
| avg | Average of elements |

## 4.2 Code/ Pseudo Code

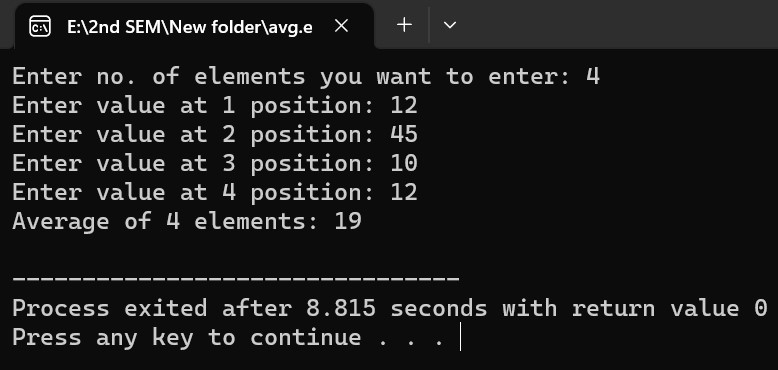
|  |
| --- |
| #include<iostream> using namespace std;  int main()  {  int i,n,a[i],avg,sum=0;  cout<<"Enter no. of elements you want to enter: ";  cin>>n;  for(i=0;i<n;i++)  {  cout<<"Enter value at "<<i+1<<" position: ";  cin>>a[i];  }  for(i=0;i<n;i++)  {  sum=sum+a[i];  }  cout<<"Average of "<<n<<" elements: "<<sum/n<<endl; return 0;  } |

## 4.3 Results

**4.3.1 Test Case 1:**



**4.3.2 Test case 2:**



### 4.3.3 Result Analysis

4.3.3.1 Advantages: printing average of an array. 4.3.3.2 Issues: N.A

# Practical 4

1. **Title: -** Write a program to insert an element at a specific position in an array.
2. **Outcome: -**inserting element at a specific position.
3. **Objectives: -** Understand the concept of Arrayinsertion and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

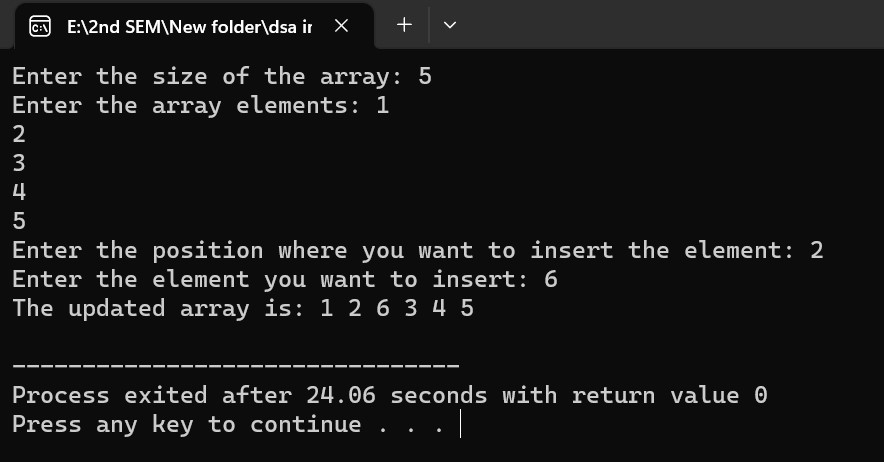
|  |  |
| --- | --- |
| arr | Name of an array |
| i | Counter for loop |
| n | No. of element |
| pos | Position to insert the element |
| elem | Element to insert in the array |

## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <iostream>  using namespace std;    int main() { const int MAX\_SIZE = 100;  int arr[MAX\_SIZE], n, pos, elem;    cout << "Enter the size of the array: "; cin >> n;    cout << "Enter the array elements: "; for (int i = 0; i < n; i++) { cin >> arr[i];  }    cout << "Enter the position where you want to insert the element: "; cin >> pos;  cout << "Enter the element you want to insert: "; cin >> elem;    for (int i = n; i > pos; i--) {  arr[i] = arr[i - 1]; } |
| arr[pos] = elem;    n++;    cout << "The updated array is: ";  for (int i = 0; i < n; i++) { cout << arr[i] << " ";  }  cout << endl;    return 0;  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: understand how to insert element at a specific position in an array.

4.3.2.2 Issues: N.A

# Practical 5

1. **Title: -** Write a program to delete an element at a specific position in an array.
2. **Outcome: -**Deleting element at a specific position.
3. **Objectives: -** Understand the concept of Arraydeletionand variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

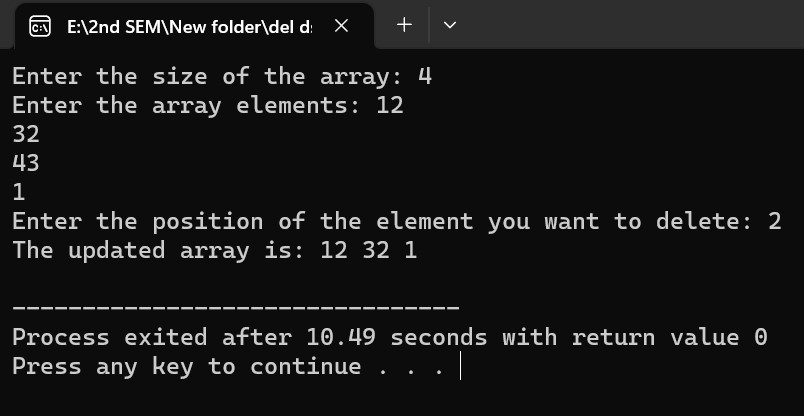
|  |  |
| --- | --- |
| arr | Name of an array |
| i | Counter for loop |
| n | No. of element |
| pos | Position to insert the element |

## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <iostream> using namespace std;    int main() { const int MAX\_SIZE = 100;  int arr[MAX\_SIZE], n, pos;    cout << "Enter the size of the array: "; cin >> n;    cout << "Enter the array elements: ";  for (int i = 0; i < n; i++) { cin >> arr[i];  }    cout << "Enter the position of the element you want to delete: "; cin >> pos;    for (int i = pos; i < n - 1; i++) {  arr[i] = arr[i + 1];  }  n--;    cout << "The updated array is: "; |
| for (int i = 0; i < n; i++) { cout << arr[i] << " ";  }  cout << endl;    return 0;  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: understand how to delete element at a specific position in an array.

4.3.2.2 Issues: N.A.

# Practical 6

1. **Title: -** Write a program to to perform linear search.
2. **Outcome: -**searching element in an array through linear search.
3. **Objectives: -** Understand the concept of searching element in an Array and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

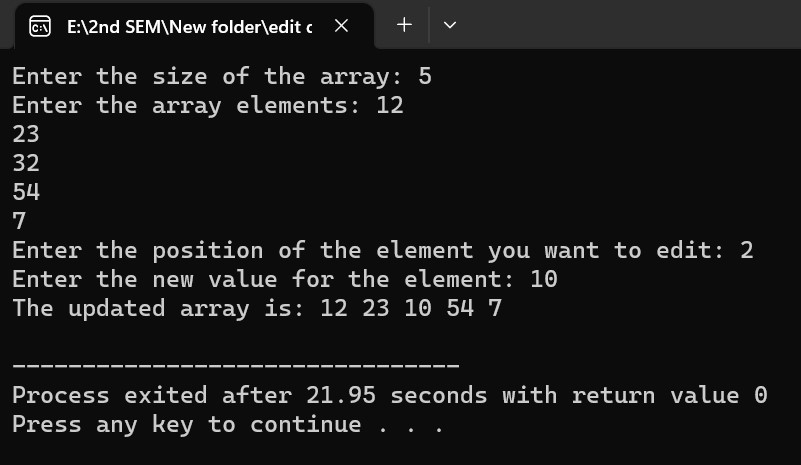
|  |  |
| --- | --- |
| a | Name of an array |
| i | Counter for loop |
| n | No. of elements |
| x | Element to be searched |
| f | Flag bit |

## 4.2 Code/ Pseudo Code

|  |  |
| --- | --- |
| #include<iostream> using namespace std; int main()  {  int a[100],i,n,x,f=0; cout<<"Enter no. of elements: ";  cin>>n;  cout<<"Enter array elements: "<<endl;  for(i=0;i<n;i++)  {  cout<<"Enter value at "<< i+1 <<" position: " ;  cin>>a[i];  }  cout<<"Entered elements: "<<endl;  for(i=0;i<n;i++)  {  cout<<a[i]<<" ";  }  cout<<"\n"<<"Enter the element to be searched in the array:  "<<endl;  cin>>x;  for(i=0;i<n;i++)  { | |
| } | if(a[i]==x)  { f=1;  break;  }  } if(f==1)  {  cout<<"Element found at "<<i+1<<" position. "<<endl;  } else  {  cout<<"Element not found."<<endl;  } return 0; |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

**4.3.2.**1Advantages: understand how to search element in an array through linear search.

4.3.2.2 Issues: N.A

# Practical 7

1. **Title: -** Write a program for insertion of element in linked list.
2. **Outcome: -**inserting a element in the linked list.
3. **Objectives: -** Understand the concept of inserting element in the linked list and variables in

C++.

1. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

|  |  |
| --- | --- |
| data | Data of node |
| add | Pointer |
| ch | choice |
| n | New element |
| New,next,prev,temp | pointer |

## 4.2 Code/ Pseudo Code

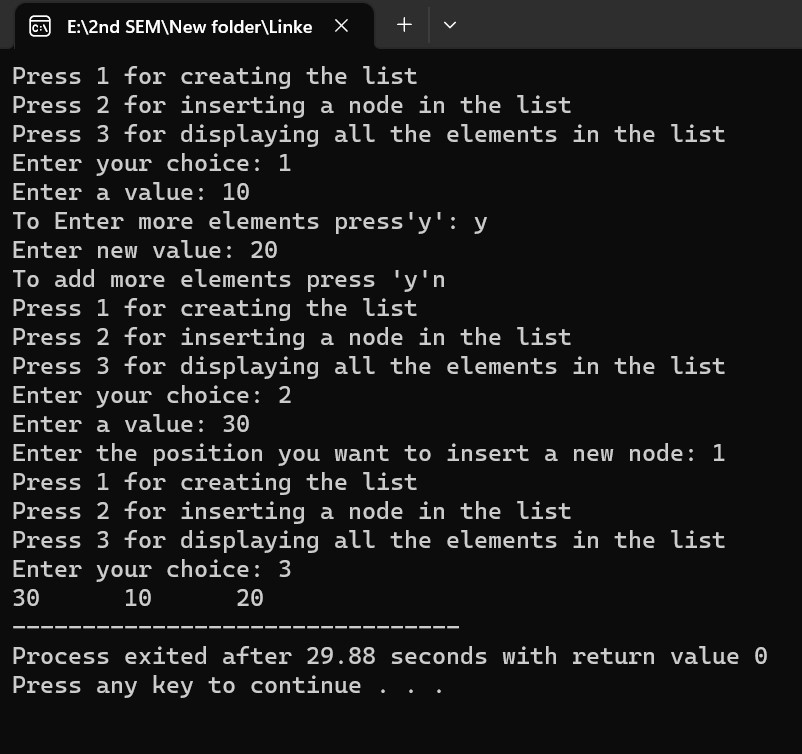
|  |
| --- |
| #include<stdio.h> #include<malloc.h> struct node  {  int data;  struct node \*add;  };  struct node \*start=NULL,\*temp,\*next,\*prev,\*New; void create(); void insert(); void display(); int main()  { int ch; do  {  printf("Press 1 for creating the list\n"); printf("Press 2 for inserting a node in the list\n"); printf("Press 3 for displaying all the elements in the list\n"); printf("Enter your choice: "); scanf("%d",&ch); |

|  |
| --- |
| switch(ch)  {  case 1:create(); break; case 2:insert(); break; case 3:display(); break;  default:printf("Invalid choice, please re-enter the choice\n");  }  }  while(ch!=3); return 0;  }  void create()  {  int n;  char ch;  printf("Enter a value: "); scanf("%d",&n);  start=(struct node \*)malloc(sizeof(struct node)); start->data=n; start->add=NULL;  temp=start;  printf("To Enter more elements press'y': "); scanf(" %c",&ch); while(ch=='Y'||ch=='y')  {  printf("Enter new value: "); scanf("%d",&n);  New=(struct node \*)malloc(sizeof(struct node));  New->data=n; New->add=NULL; temp->add=New; temp=New;  printf("To add more elements press 'y'");  scanf(" %c",&ch);  }  }  void display()  {  if (start==NULL)  {  printf("List not created,create a list first");  }  else  {  temp=start; while(temp!=NULL) |

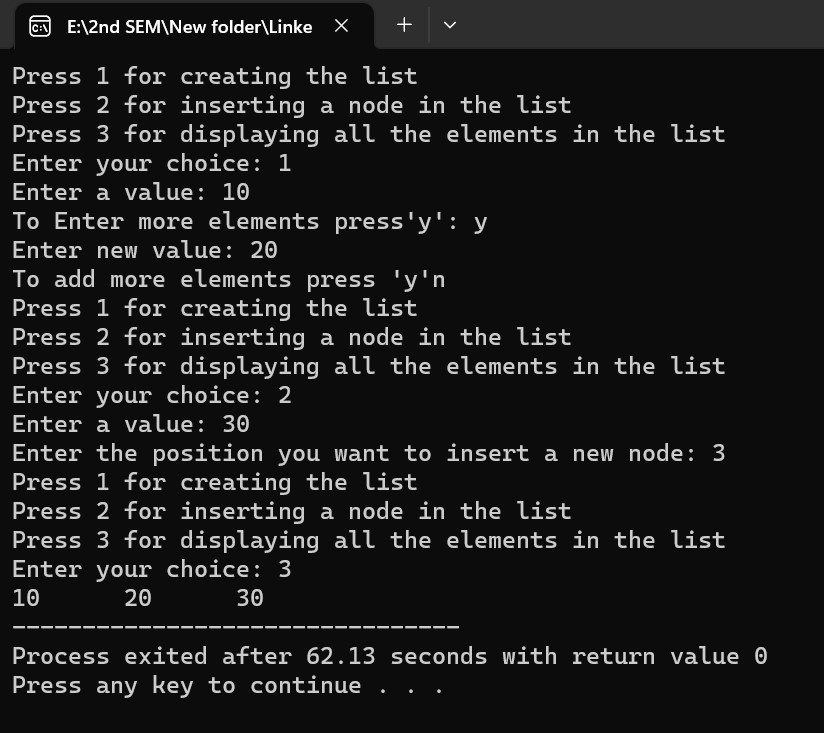
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| {  printf("%d\t",temp->data); temp=temp->add;  }  }  }  void insert()  {  int n,pos,count=0,i=1; if(start==NULL)  {  printf("List not created, create a list first.");  }  else  {  printf("Enter a value: "); scanf("%d",&n);  New=(struct node \*)malloc(sizeof(struct node));  New->data=n;  New->add=NULL;  printf("Enter the position you want to insert a new node: ");  scanf("%d",&pos);  if(pos==1) // AT FIRST POSITION  {  New->add=start;  start=New;  }  else  {  temp=start;  while(temp!=NULL)  {  count++;  temp=temp->add;  }  if(pos>count+1) // AT LAST POSITION  {  printf("Invalid position,it should be greater than equal to %d",count+1);  }  else if(pos==count+1)  {  temp=start;  while(temp->add!=NULL)  {  temp=temp->add;  } | | | | | |
|  |  |  |  |  | temp->add=New; |
|  |  |  |  | } else  { | next=start;  while(i<pos) // AT SPECIFIC POSITION  {  prev=next;  next=next->add;  i++;  }  prev->add=New; New->add=next; |
|  |  | } |  | } |  |
| } | } |  |  |  |  |

## 4.3 Results

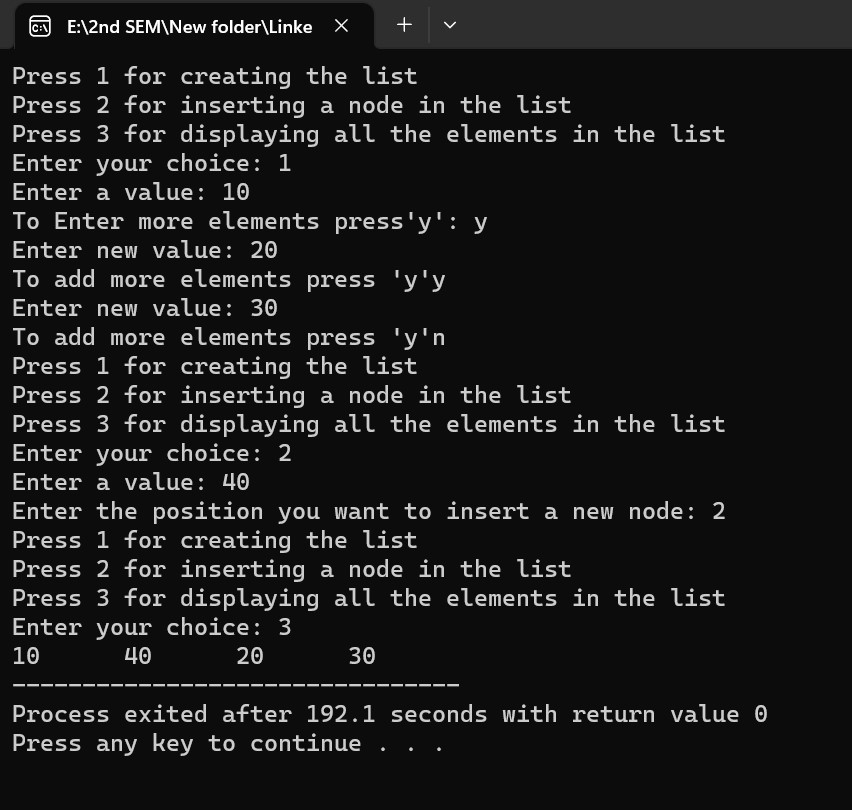
**4.3.1 Test Case 1: inserting element at first position.**



**4.3.2 Test case 2: inserting element at last position.**



**4.3.3 Test case 3: inserting element at specific position**



**4.3.4**  **Result Analysis:**

**4.3.4.1** Advantages: understand how to insert element in the

linked list

**4.3.4.2** Issues: N.A

# Practical 8

1. **Title:** Write a program for deletion of element in linked list.
2. **Outcome:** deleting a element in the linked list.
3. **Objective:** Understand the concept of deleting element in the linked list and variables in C++.

## 4. Nomenclature, theory with self-assessment questionnaire: -

**4.1 Nomenclature:**

|  |  |
| --- | --- |
| data | Data of node |
| add | Pointer |
| ch | choice |
| n | New element |
| New,next,prev,temp | pointer |

### 4.2 Code/ Pseudo Code

|  |
| --- |
| #include<stdio.h> #include<malloc.h>  struct node  {  int data; struct node \*add;  };  struct node \*start=NULL,\*temp,\*next,\*prev,\*New; void create(); void insert(); void Delete(); void display(); int main()  { int ch; do  {  printf("Press 1 for creating the list\n"); printf("Press 2 for inserting a node in the list\n"); printf("Press 3 for deleting a node in the list\n"); |

printf("Press 4 for displaying all the elements in the list\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:create(); break; case 2:insert(); break; case 3:Delete(); break;

case 4:display(); break;

default:printf("Invalid choice, please re-enter the choice\n");

}

} while(ch!=4); return 0; } void create()

{

int n; char ch;

printf("Enter a value: "); scanf("%d",&n);

start=(struct node \*)malloc(sizeof(struct node)); start->data=n; start->add=NULL;

temp=start;

printf("To Enter more elements press'y': "); scanf(" %c",&ch);

while(ch=='Y'||ch=='y')

{

printf("Enter new value: "); scanf("%d",&n);

New=(struct node \*)malloc(sizeof(struct node));

New->data=n; New->add=NULL; temp->add=New;

temp=New;

printf("To add more elements press 'y'"); scanf(" %c",&ch);

}

}

void display()

{

if (start==NULL)

{

printf("List not created,create a list first");

}

else

{

temp=start;

while(temp!=NULL)

{

printf("%d\t",temp->data); temp=temp->add;

}

}

}

void insert()

{

int n,pos,count=0,i=1; if(start==NULL)

{

printf("List not created, create a list first.");

}

else

{

printf("Enter a value: "); scanf("%d",&n);

New=(struct node \*)malloc(sizeof(struct node));

New->data=n;

New->add=NULL;

printf("Enter the position you want to insert a new node: ");

scanf("%d",&pos); if(pos==1)

{

New->add=start;

start=New;

}

else

{

temp=start; while(temp!=NULL)

{

count++;

temp=temp->add;

}

if(pos>count+1)

{

printf("Invalid position,it should be greater than equal to %d",count+1);

}

else if(pos==count+1)

{

temp=start;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | while(temp->add!=NULL) |
|  |  |  |  | { |
|  |  |  |  | temp=temp->add; |
|  |  |  |  | } |
|  |  |  |  | temp->add=New; |
|  |  |  | } |  |
|  |  |  | else |  |
|  |  |  | { |  |
|  |  |  |  | next=start; |
|  |  |  |  | while(i<pos) |
|  |  |  |  | { |
|  |  |  |  | prev=next; |
|  |  |  |  | next=next->add; |
|  |  |  |  | i++; |
|  |  |  |  | } |
|  |  |  |  | prev->add=New; |
|  |  |  |  | New->add=next; |
|  |  |  | } |  |
|  |  | } |  |  |
| } | } |  |  |  |
| void Delete() { | |
| int count=0,i=1,position; | | | |

if(start == NULL)

{

printf("List Not Created, Create a list first");

} else

{

printf("Enter a Posiotion for Deletion: "); scanf("%d",&position);

if(position == 1) // deleting from first position

{ temp = start;

start = start->add;

printf("Deleted Element = %d: \n",temp->data); free(temp);

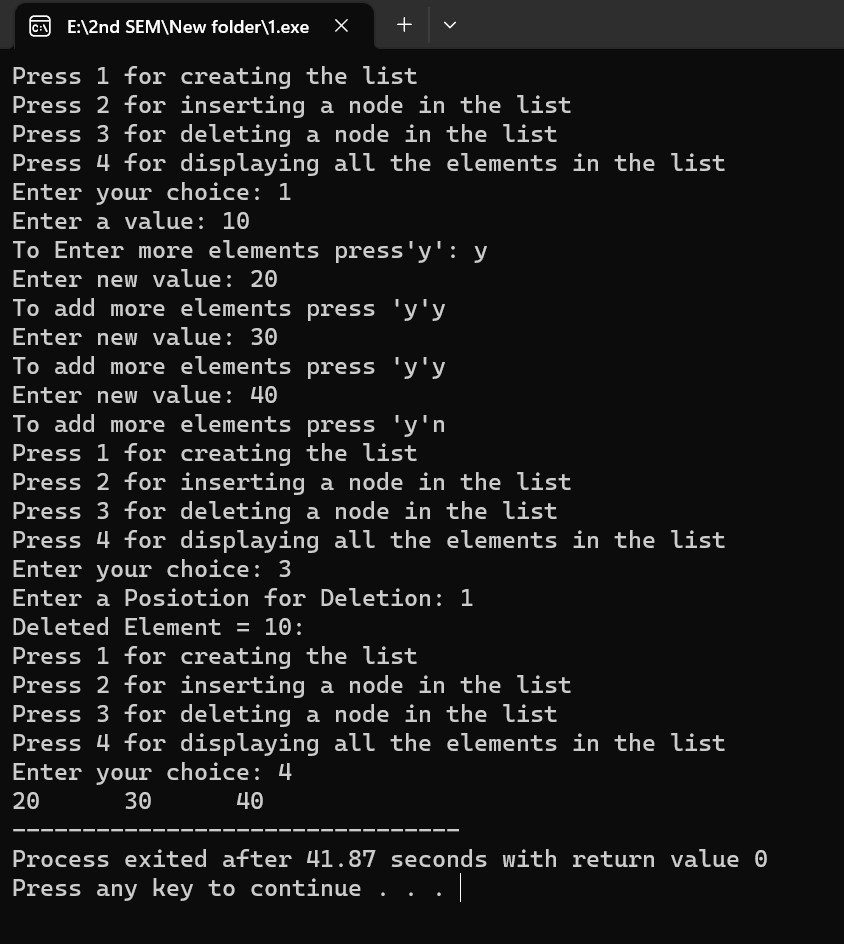
} else { temp = start;

while(temp!=NULL)

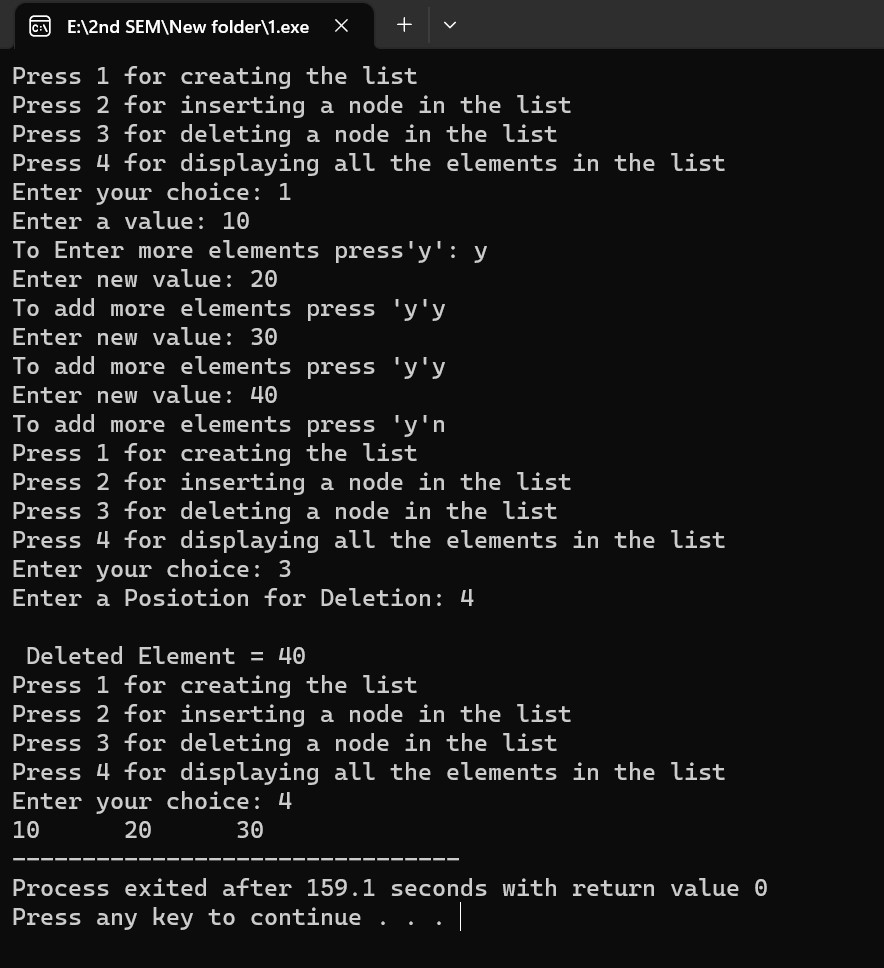
|  |
| --- |
| {  count++;  temp = temp->add;  }  if(position > count)  {  printf("\nInvalid Position, it can't be more than %d\n",count);  }  else if(position == count) // deleting from Last Position  {  next = start;  while(next->add!=NULL)  {  prev=next;  next = next->add;  }  printf("\n Deleted Element = %d\n",next->data); free(next);  prev->add = NULL;  } else {  next = start;  while(i<position) // deleting at specific position  { prev = next;  next = next->add;  i++;  }  temp = next; next = next->add;  printf("\nDeleted Element = %d\n",temp->data); free(temp); prev->add=next;  }  }  }    } |

### 4.3 Results

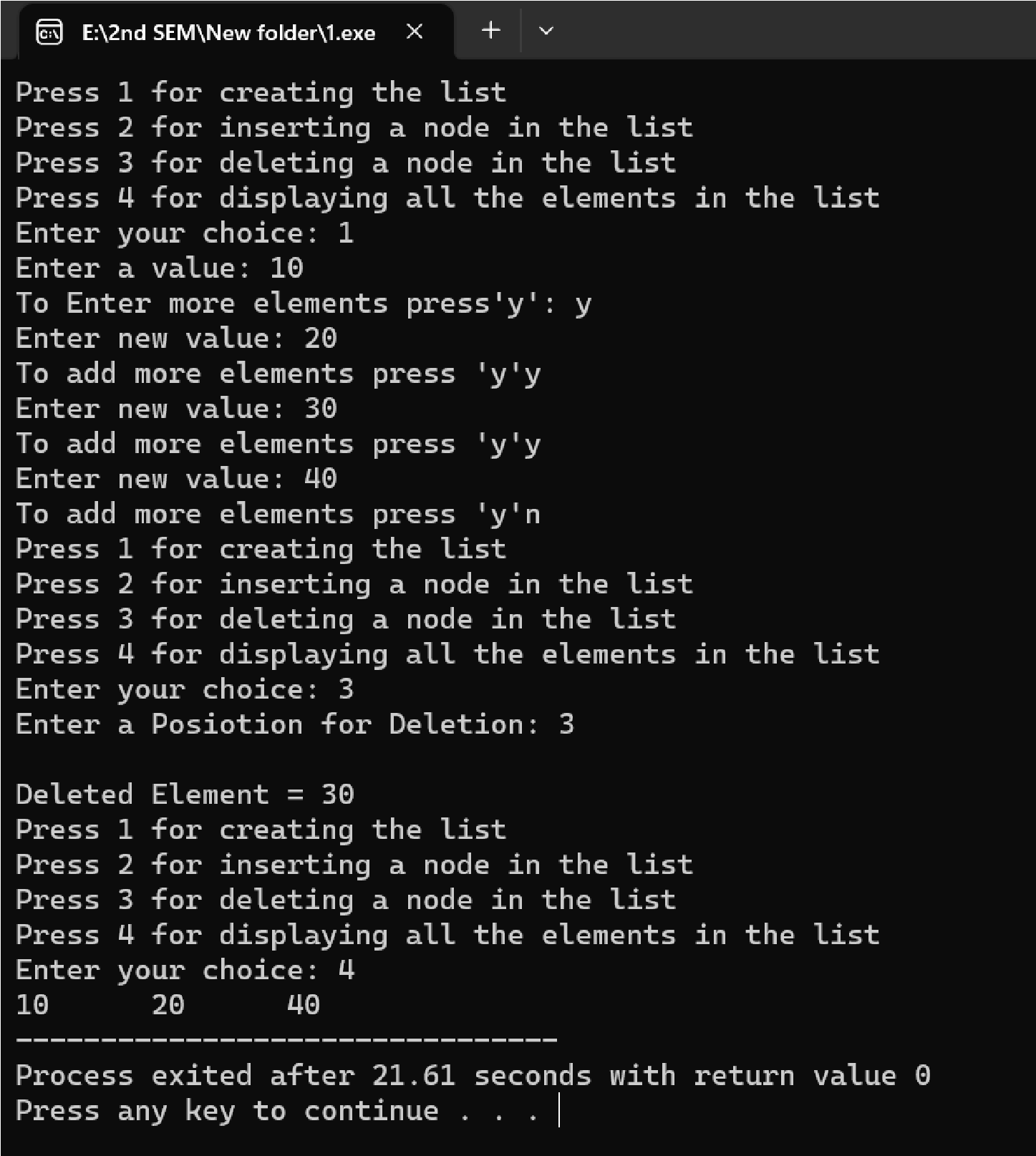
**4.3.1 Test Case 1: deleting element at first position.**



**4.3.2 Test case 2: deleting element at last position.**



**4.3.3 Test case 3: deleting element at specific position**



#### 4.3.4 Result Analysis

4.3.4.1 Advantages: understand how to delete element in the linked list

4.3.4.2 Issues: N.A

# Practical 9

1. **Title: -** Write a program to insert an element in stack.
2. **Outcome: -** element inserting in form of stack.
3. **Objectives: -** Understand the concept of stack and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

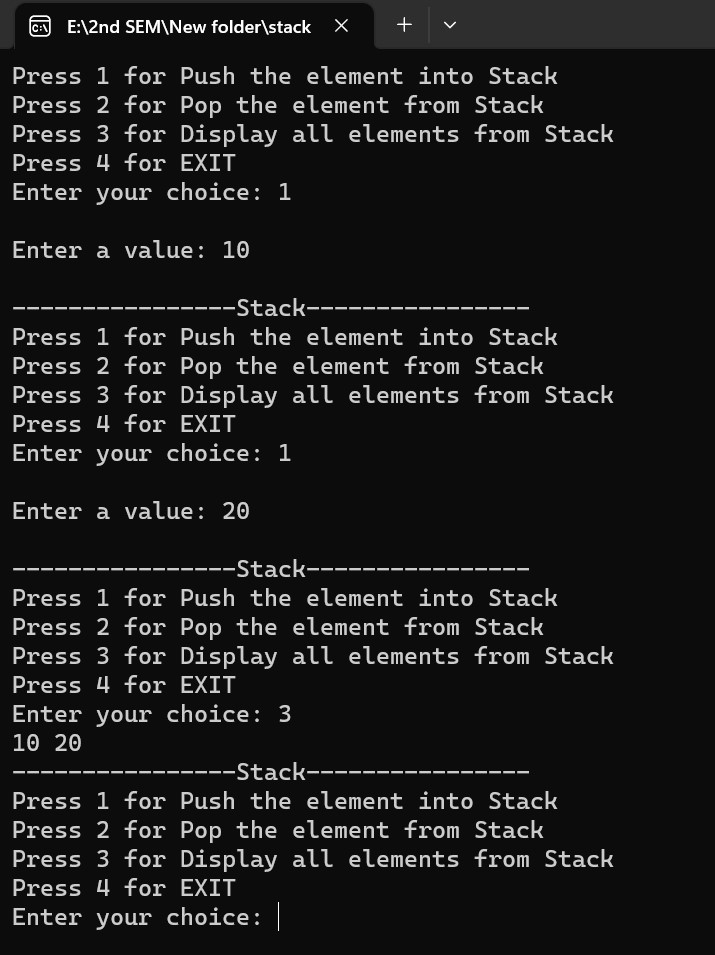
|  |  |
| --- | --- |
| maxsize | Maximum size 10 |
| top | -1 |
| n | No. of elements |
| ch | Choice |
| i | Counter for loop |

## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <stdio.h>    #define maxsize 10    int stack[maxsize], top = -1;    void push(); void display();    int main()  {  int ch;    do {  printf("\n----------------Stack----------------\n"); printf("Press 1 for Push the element into Stack\n"); printf("Press 2 for Pop the element from Stack\n"); printf("Press 3 for Display all elements from Stack\n"); printf("Press 4 for EXIT\n"); printf("Enter your choice: "); scanf("%d",&ch); |
| switch(ch)  {  case 1:push();break; case 3:display();break; case 4:break;  default:printf("Invalid Choice, re-enter the choice again: ");  }  }  while(ch!=4);    return 0;  }    void push()  {  int n;  if(top == (maxsize-1))  {  printf("\nStack Overflow..\n");  } else {  printf("\nEnter a value: ");  scanf("%d",&n);  top++;  stack[top]=n;  } }  void display()  {  int i;  if(top == -1)  {  printf("\nStack Underflow\n");  } else {  for (i = 0; i <= top; i++)  {  printf("%d ",stack[i]);  }  }  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: performing push operation in stack.

4.3.2.2 Issues: N.A

# Practical 10

1. **Title: -** Write a program to delete an element in stack.
2. **Outcome: -** element deleting in form of stack.
3. **Objectives: -** Understand the concept of stack and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

|  |  |
| --- | --- |
| maxsize | Maximum size 10 |
| top | -1 |
| n | No. of elements |
| ch | Choice |
| i | Counter for loop |

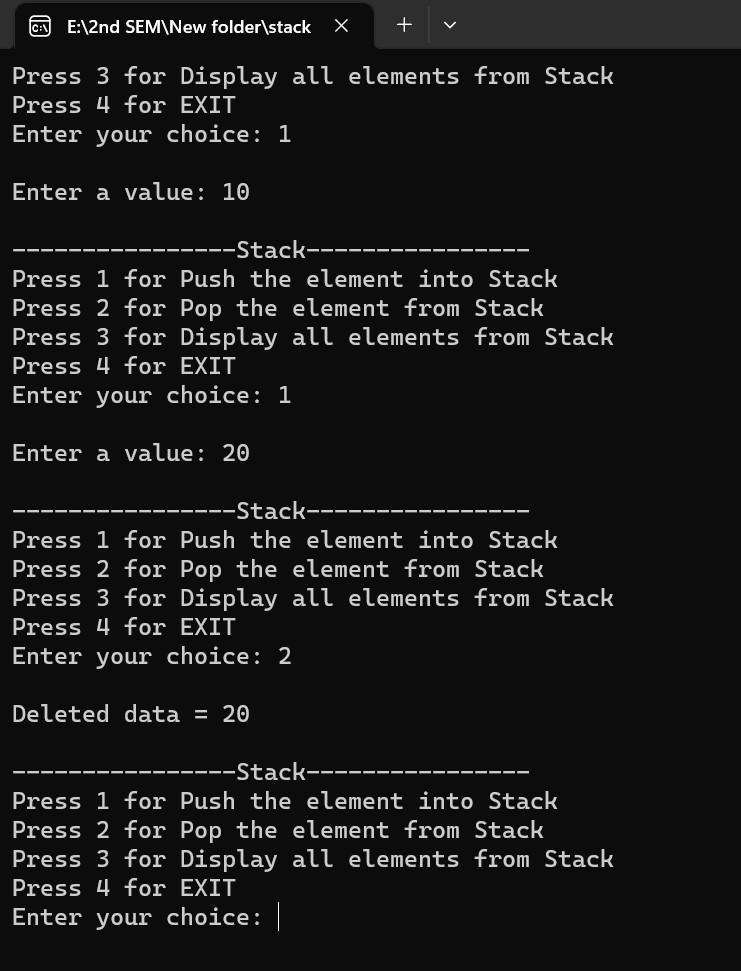
## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <stdio.h>    #define maxsize 10    int stack[maxsize], top = -1;    void push(); void pop();  void display();    int main()  {  int ch;    do {  printf("\n----------------Stack----------------\n"); printf("Press 1 for Push the element into Stack\n"); printf("Press 2 for Pop the element from Stack\n"); printf("Press 3 for Display all elements from Stack\n"); printf("Press 4 for EXIT\n"); printf("Enter your choice: "); |

|  |
| --- |
| scanf("%d",&ch);    switch(ch)  {  case 1:push();break; case 2:pop(); break; case 3:display();break; case 4:break;  default:printf("Invalid Choice, re-enter the choice again: ");  }  }  while(ch!=4);    return 0;  }    void push()  {  int n;  if(top == (maxsize-1))  {  printf("\nStack Overflow..\n");  } else {  printf("\nEnter a value: ");  scanf("%d",&n); top++;  stack[top]=n;  }  }    void pop()  {  if(top == -1)  {  printf("\nStack Underflow\n");  } else  {  printf("\nDeleted data = %d\n",stack[top]); top--;  }  }    void display() |
| {  int i;  if(top == -1)  {  printf("\nStack Underflow\n");  } else {  for (i = 0; i <= top; i++)  {  printf("%d ",stack[i]);  }  }  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

**4.3.2.1** Advantages: performing pop operation in stack.

**4.3.2.2**  Issues: N.A

# Practical 11

1. **Title: -** Write a program to insert an element in queue.
2. **Outcome: -** element inserting in form of queue.
3. **Objectives: -** Understand the concept of queue and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

|  |  |
| --- | --- |
| maxsize | Maximum size 10 |
| i | Counter for loop |
| n | No. of elements |
| rear | -1 |
| front | 0 |
| ch | Choice |

## 4.2 Code/ Pseudo Code

#include <stdio.h>

#define maxsize 10

int q[maxsize], rear = -1, front = 0;

void insert();

void display();

int main() { int ch;

do

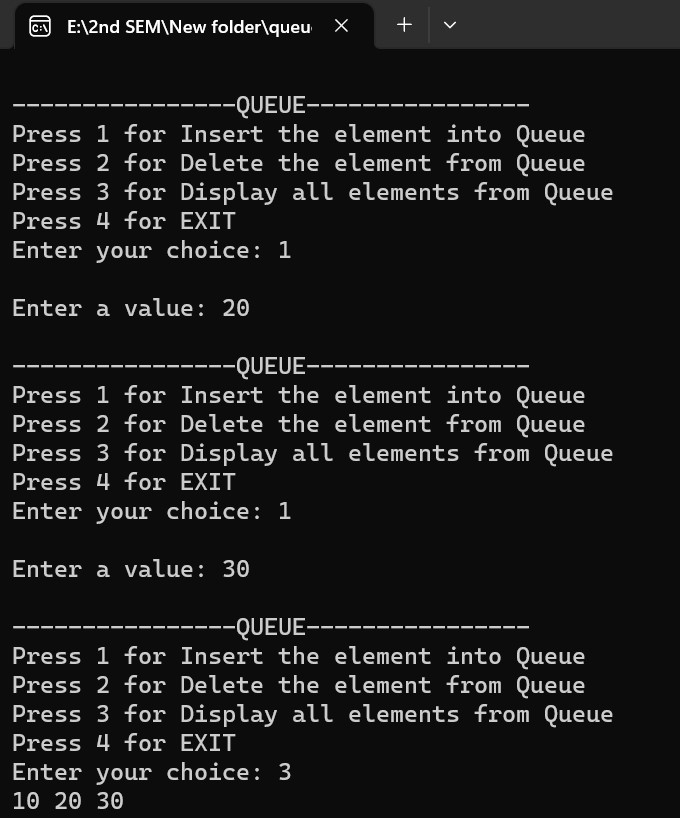
{

printf("\n----------------QUEUE----------------\n"); printf("Press 1 for Insert the element into Queue\n"); printf("Press 2 for Delete the element from Queue\n"); printf("Press 3 for Display all elements from Queue\n");

|  |
| --- |
| printf("Press 4 for EXIT\n"); printf("Enter your choice: "); scanf("%d",&ch);    switch(ch)  {  case 1:insert();break; case 3:display();break; case 4:break;  default:printf("Invalid Choice, re-enter the choice again: ");  }  }  while(ch!=4);    return 0;  }  void insert()  {  int n;  if(rear == (maxsize-1))  {  printf("\nQueue Overflow..\n");  } else {  printf("\nEnter a value: ");  scanf("%d",&n); rear++; q[rear]=n;  } }  void display()  {  int i;  if(rear<front)  {  printf("\nQueue Underflow\n");  } else {  for (i = front; i <= rear; i++)  {  printf("%d ",q[i]);  }  }  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

**4.3.2.3** Advantages: performing insertion in queue.

**4.3.2.4**  Issues: N.A

# Practical 12

1. **Title: -** Write a program to delete an element in queue.
2. **Outcome: -** element deleting in form of queue.
3. **Objectives: -** Understand the concept of queue and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

|  |  |
| --- | --- |
| a | Name of an array |
| i | Counter for loop |
| n | No. of elements |
| sum | Sum of elements |
| avg | Average of elements |

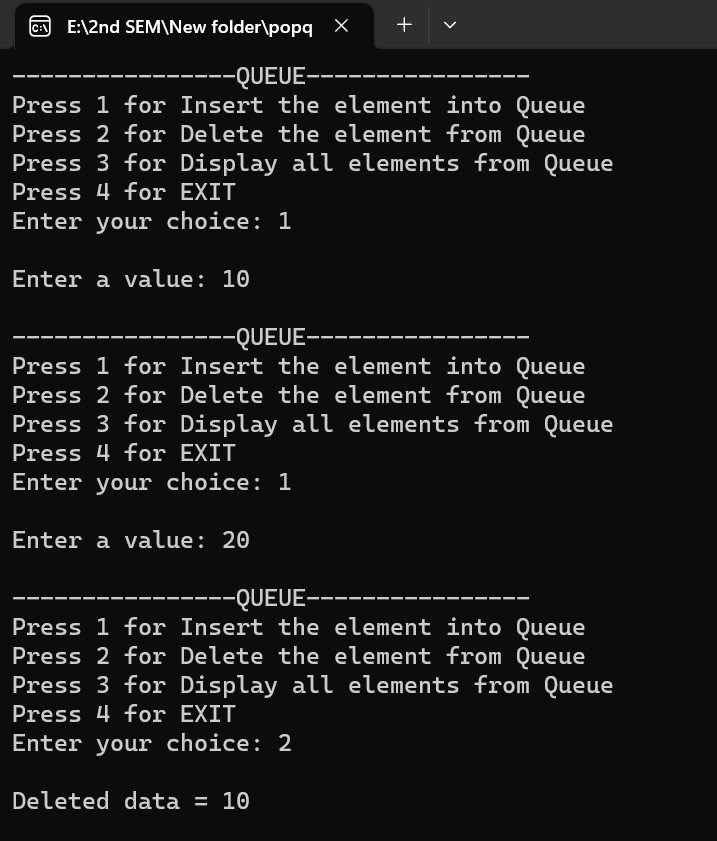
## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <stdio.h>    #define maxsize 10    int q[maxsize], rear = -1, front = 0;    void insert(); void Delete();  void display();    int main()  {  int ch;    do  {  printf("\n----------------QUEUE----------------\n"); printf("Press 1 for Insert the element into Queue\n"); printf("Press 2 for Delete the element from Queue\n"); printf("Press 3 for Display all elements from Queue\n"); printf("Press 4 for EXIT\n"); printf("Enter your choice: "); |

|  |
| --- |
| scanf("%d",&ch);    switch(ch)  {  case 1:insert();break; case 2:Delete(); break; case 3:display();break; case 4:break;  default:printf("Invalid Choice, re-enter the choice again: ");  }  }  while(ch!=4);    return 0;  }  void insert()  {  int n;  if(rear == (maxsize-1))  {  printf("\nQueue Overflow..\n");  } else {  printf("\nEnter a value: "); scanf("%d",&n);  rear++; q[rear]=n;  } }  void Delete()  {  if(rear<front)  {  printf("\nQueue Underflow\n");  } else  {  printf("\nDeleted data = %d\n",q[front]); front++;  } }  void display()  {  int i; if(rear<front) |
| {  printf("\nQueue Underflow\n");  } else {  for (i = front; i <= rear; i++)  {  printf("%d ",q[i]);  }  }  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: performing deletion in queue.

**4.3.2.2** Issues: N.

# Practical 13

1. **Title: -** Write a program to perform binary search.
2. **Outcome: -** searching element in the given data through binary search.
3. **Objectives: -** Understand the concept of binary search and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

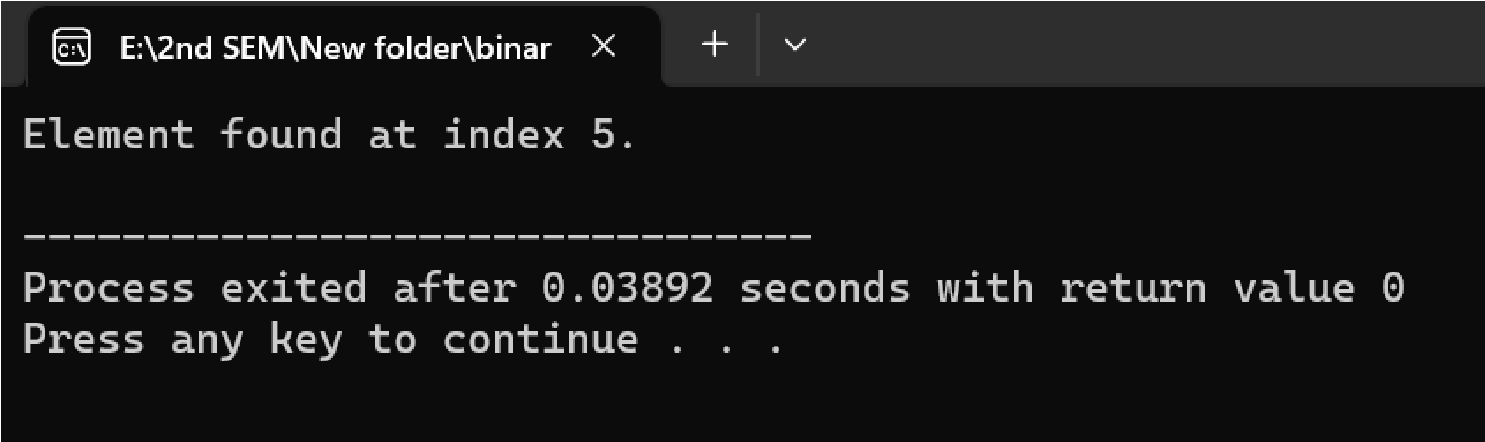
|  |  |
| --- | --- |
| arr | Name of an array |
| i | Counter for loop |
| n | No. of elements |
| mid,left,right | Variables |
| result | End resulr |

## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <stdio.h>    int binarySearch(int arr[], int left, int right, int key) { while (left <= right) {  int mid = left + (right - left) / 2;    if (arr[mid] == key) return mid;    if (arr[mid] < key) left = mid + 1; else  right = mid - 1;  }    return -1; // key not found  }    int main() {  int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}; int n = sizeof(arr) / sizeof(arr[0]); int key = 23; |
| int result = binarySearch(arr, 0, n - 1, key);    if (result == -1)  printf("Element not found in the array.\n"); else  printf("Element found at index %d.\n", result);    return 0;  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: performing binary search to search the required element .

**4.3.2.2** Issues: N.A

# Practical 14

1. **Title: -** Write a program to perform bubble sort.
2. **Outcome: -** sort the element through bubble sort.
3. **Objectives: -** Understand the concept of sorting and variables in C+ +.
4. **Nomenclature, theory with self-assessment questionnaire: - 4.1 Nomenclature:**

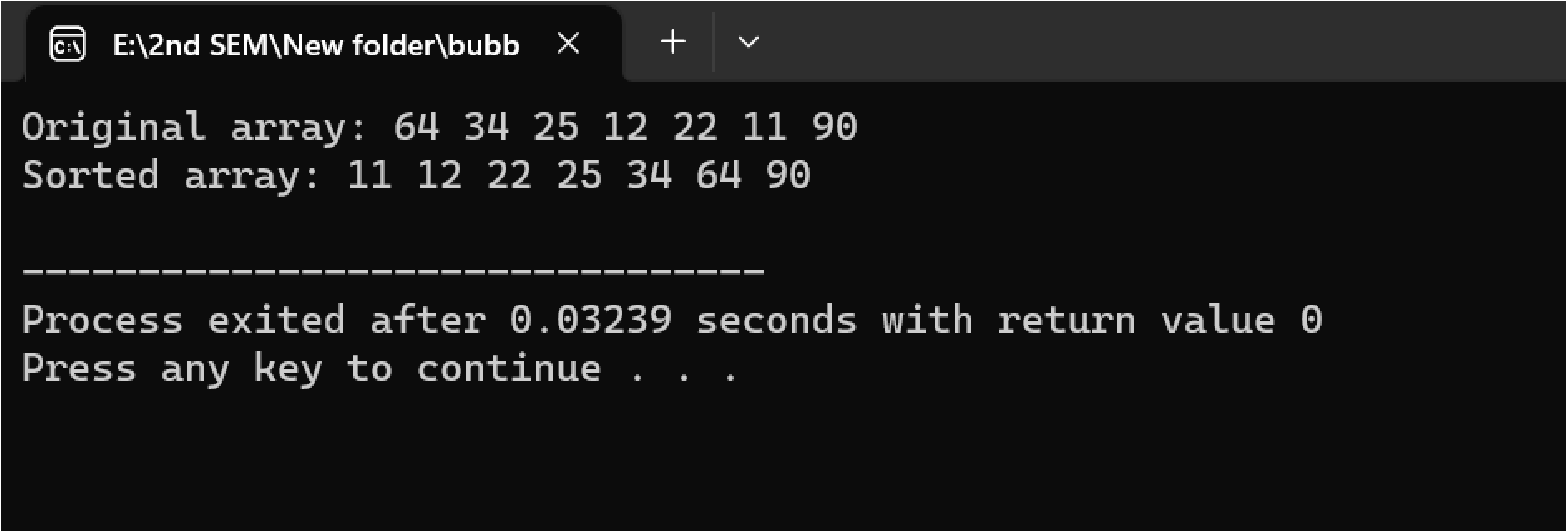
|  |  |
| --- | --- |
| arr | Name of an array |
| i | Counter for loop |
| j | Counter for loop |
| temp | Temporary variable |

## 4.2 Code/ Pseudo Code

|  |
| --- |
| #include <stdio.h>    void bubbleSort(int arr[], int n) { for (int i = 0; i < n - 1; i++) { for (int j = 0; j < n - i - 1; j++) { if (arr[j] > arr[j + 1]) {  // Swap arr[j] and arr[j + 1] int temp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = temp;  }  }  }  }    void printArray(int arr[], int n) { for (int i = 0; i < n; i++) printf("%d ", arr[i]); printf("\n");  }    int main() { int arr[] = {64, 34, 25, 12, 22, 11, 90}; int n = sizeof(arr) / sizeof(arr[0]); |
| printf("Original array: "); printArray(arr, n);    bubbleSort(arr, n);    printf("Sorted array: ");  printArray(arr, n);    return 0;  } |

## 4.3 Results

**4.3.1 Test Case:**



### 4.3.2 Result Analysis

4.3.2.1 Advantages: printing average of an array.

**4.3.2.2** Issues: N.A