Program 1. WAP to create a linear array named LA of size 6,and perform traversing operation on it using function name "Traverse".

Code:

```
#include <stdio.h>
#include "intro.c"
void traverse(int *arr, int n) {
  for (size_t i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
}

int main() {
  printIntro("creating and traversing an array");
  int n = 6;
  int LA[] = {4, 5, 7, 8, 3, 9};

  printf("array created of size %d\nenter item to be stored\n",n) ;
  for(int i = 0 ; i < n ; i++){
        printf("enter item %d : ",(i+1)) ;
        scanf("%d",&LA[i]) ;
  }
  printf("Traversing array...\n");
  traverse(LA, n);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : creating and traversing an array

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
array created of size 6
enter item to be stored
enter item 1 : 12
enter item 2 : 34
enter item 3 : 45
enter item 4 : 67
enter item 5 : 78
enter item 6 : 89
Traversing array...
12 34 45 67 78 89
```

Program 2. WAP to insert an element from a linear array using function name "Insert".

Code:

```
#include <stdio.h>
 #include "intro.c"

void traverse(int *arr, int n) {
  for (size_t i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
}

void insert(int *arr, int n, int *indexOfLastItem) {
  for (size_t i = 0; i < n; i++) {
    arr[i] = i * 3;
    *indexOfLastItem += 1;
  }
}

void insertAtPosition(int *arr, int position, int n, int *indexOfLastItem) {
  if (*indexOfLastItem >= n - 1) {
    printf("no more items can be inserted\n ");
    return;
  }

  if (*indexOfLastItem != -1  ) {
    // shift items backward
    for (size_t i = *indexOfLastItem; i >= position; i--) {
      arr[i + 1] = arr[i];
    }
  }
  printf("Enter element to be inserted : ");
  scanf("%d", &arr[position]);
  *indexOfLastItem += 1;
}
```

```
int main() {
  printIntro("inserting elements to array using function");
  int n = 10;
  int indexOfLastItem = -1;
  int LA[n];
  int position;

  insert(LA, 6, &indexOfLastItem);
  printf("Array elements :\n");
  traverse(LA, indexOfLastItem);

  printf("\nenter position : ");
  scanf("%d", &position);
  insertAtPosition(LA, position, 10, &indexOfLastItem);
  printf("Array elements :\n");
  traverse(LA, indexOfLastItem);

  return 0;
}
```

Output:

```
Author : Jitendra Kumar Sahu
Program topic : inserting elements to array using function
Array elements :
0 3 6 9 12
enter position : 3
Enter element to be inserted : 22222
Array elements :
0 3 6 22222 9 12
```

Program 3. WAP to delete an element from a linear array using function name "delete".

Code:

```
#include <stdio.h>
#include "intro.c"
void deleteFromPosition(int *arr, const int position, int
*indexOfLastItem) {
  if (position < 0 || position > *indexOfLastItem) {
    printf("\nCan not be deleted!");
    return;
  }
  printf("\ndeleted item : %d\n", arr[position]);
  // move items forward
  for (size_t i = position; i < *indexOfLastItem; i++) {
    arr[i] = arr[i + 1];
  }
  *indexOfLastItem -= 1;
}
int main() {
  printIntro("Deleting item from given postion in array");

  int LA[10] = {23, 56, 8, 98, 5, 3, 687, 5};
  int indexOfLastItem = 7;

  printf("Array elements are : \n");
  for (size_t i = 0; i < indexOfLastItem; i++) printf("%d ", LA[i]);

  int position;
  printf("\nenter position item to be deleted from : ");
  scanf("%d", &position);
  deleteFromPosition(LA, position, &indexOfLastItem);
  for (size_t i = 0; i < indexOfLastItem; i++) printf("%d ", LA[i]);
 return 0;
}
```

Output:

```
C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>
Author : Jitendra Kumar SAHU
Program Topic : Deleting item from given postion in array

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Array elements are :
23 56 8 98 5 3 687
enter position item to be deleted from : 2

deleted item : 8
23 56 98 5 3 687
C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>
```

Program 4. WAP to create a single node in linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node ;

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

int main() {
  printIntro("creating a single node in linked list.");

  node *START = NULL ;

  node *loc = (node*)malloc(sizeof(node)) ;
  loc->data = 50 ;
  loc->next = NULL  ;
  START = loc ;

  printf("data at node : %d\n",START->data);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : creating a single node in linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
data at node : 50
```

Program 5. WAP to create linked list at compile time having 4 node.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc)
  { printf("no node exist!\n") ;
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
}
```

```
int main() {
  printIntro("linked list having 4 node at compile time");

  node *start = NULL;
  // insertion of first node
  node *loc = getNode(1);
  start = loc;

  loc->next = getNode(2);
  loc = loc->next;

  loc->next = getNode(3);
  loc = loc->next;

  loc->next = getNode(4);
  loc = loc->next;

  printf("node(s) at linked list");
  traverse(start);

  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : linked list having 4 node at compile time

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
node(s) at linked list
|1|D02928| -> |2|D02938| -> |3|D02948| -> |4|0|
```

Program 6. WAP to create linked list having 4 node at runtime.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"
typedef struct node node;
struct node {
  int data;
  node *next;
};

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
   node *loc = start;
  if (!loc)
  { printf("no node exist!\n") ;
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
}
```

```
int main() {
  printIntro("linked list having node(s) at runtime");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ?  : ");
  scanf("%d", &n);
  for (int data, i = 0; i < n; i++) {
    printf("Enter data for node %d : ", i + 1);
    scanf("%d", &data);
    if (start == NULL) {
      loc = getNode(data);
      start = loc;
    } else {
      loc->next = getNode(data);
      loc = loc->next;
    }
  }
  printf("nodes of linked list\n") ;
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : linked list having node(s) at runtime

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ?  : 6
Enter data for node 1 : 12
Enter data for node 2 : 34
Enter data for node 3 : 23
Enter data for node 4 : 56
Enter data for node 5 : 13
Enter data for node 6 : 54
nodes of linked list

|12|B12928| -> |34|B12938| -> |23|B12948| -> |56|B12958| -> |13|B12968| -> |54|0|
```

Program 7. WAP to perform trversing in linked list using function named "Traverse".

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc)
  { printf("no node exist!\n") ;
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
```

```
  printf("\n\n");
}

int main() {
  printIntro("traversing in linked list using function named \"Traverse()\"");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ?  : ");
  scanf("%d", &n);
  for (int data, i = 0; i < n; i++) {
    printf("Enter data for node %d : ", i + 1);
    scanf("%d", &data);
    if (start == NULL) {
      loc = getNode(data);
      start = loc;
    } else {
      loc->next = getNode(data);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n") ;
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : traversing in linked list using function named "Traverse()"

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ?  : 5
Enter data for node 1 : 12
Enter data for node 2 : 23
Enter data for node 3 : 34
Enter data for node 4 : 45
Enter data for node 5 : 56
nodes in linked list :

|12|9B2928| -> |23|9B2938| -> |34|9B2948| -> |45|9B2958| -> |56|0|
```

Program 8. WAP to insert a node at the begining of the linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
```

```
}

int main() {
  printIntro("inserting at end of the linked list.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ?  : ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    if (start == NULL) {
      loc = getNode(1);
      start = loc;
    } else {
      loc->next = getNode((i + 1) * 3);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);

  int data;
  printf("enter item to be inserted at first position : ");

// creating and inserting node at first postion
  scanf("%d", &data);
  node *newNode = getNode(data);
  newNode->next = start;
  start = newNode;

  printf("nodes in linked list after insertion: \n");
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : inserting at end of the linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ?  : 6
nodes in linked list :

|1|B12928| -> |6|B12938| -> |9|B12948| -> |12|B12958| -> |15|B12968| -> |18|0|

enter item to be inserted at first position : 1111
nodes in linked list after insertion:

|1111|B12918| -> |1|B12928| -> |6|B12938| -> |9|B12948| -> |12|B12958| -> |15|B12968| -> |18|0|
```

Program 9. WAP to insert a node at the end of the linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
```

```
}

void insertAtLastNode(node *start, int data) {
  node *loc = start;
  // get last node
  while (loc->next) loc = loc->next;
  loc->next = getNode(data);
}

int main() {
  printIntro("inserting end of the linked list.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ?  : ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    if (start == NULL) {
      loc = getNode(1);
      start = loc;
    } else {
      loc->next = getNode((i + 1) * 3);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);

  int data;
  printf("enter item to be inserted at end : ");

  // creating and inserting node at END
  scanf("%d", &data);
  if (start)
```

```
    insertAtLastNode(start, data);
  else
    start = getNode(data);

  printf("After insertion: \n");
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : inserting end of the linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ?  : 5
nodes in linked list :

|1|6E2928| -> |6|6E2938| -> |9|6E2948| -> |12|6E2958| -> |15|0|

enter item to be inserted at end : 55555
After insertion:

|1|6E2928| -> |6|6E2938| -> |9|6E2948| -> |12|6E2958| -> |15|6E2968| -> |55555|0|
```

Program 10. WAP to insert a node at the specific position of the linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
```

```
}

node *insertAtPosition(node *start, const int position, int data) {
  int currentPosition = 0;
  node *loc, *locp;
  loc = locp = start;

  while (currentPosition < position) {
    locp = loc;
    loc = loc->next;
    currentPosition++;
  }
  node *newNode = getNode(data);
  newNode->next = loc;
  locp->next = newNode;
}
int main() {
  printIntro("inserting at given postion of the linked list.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ? ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    if (start == NULL) {
      loc = getNode(1);
      start = loc;
    } else {
      loc->next = getNode((i + 1) * 3);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);
```

```
  int position;
  printf("enter position (indexing from 0) where node to be inserted : ");
  scanf("%d", &position);

  if (position < 0 || position > n) {
    printf("invalid position!");
    return 1;
  }

  int data;
  printf("enter data : ");
  scanf("%d", &data);
  if (position == 0) {
    node *newNode = getNode(data);
    newNode->next = start;
    start = newNode;
  } else {
    insertAtPosition(start, position, data);
  }

  printf("\nafter insertion : \n");
  traverse(start);

  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : inserting at given postion of the linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ? 5
nodes in linked list :

|1|32928| -> |6|32938| -> |9|32948| -> |12|32958| -> |15|0|

enter position (indexing from 0) where node to be inserted : 2
enter data : 22222

after insertion :

|1|32928| -> |6|32968| -> |22222|32938| -> |9|32948| -> |12|32958| -> |15|0|
```

Program 11. WAP to delete a node at the begining of the linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) exit(1);
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }
  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
}
```

```
int main() {
  printIntro("delete from firts postion of the linked list.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ? ");
  scanf("%d", &n);
  int data;
  for (int i = 0; i < n; i++) {
    printf("enter data for node %d : ", i + 1);
    scanf("%d", &data);
    if (start == NULL) {
      loc = getNode(data);
      start = loc;
    } else {
      loc->next = getNode(data);
      loc = loc->next;
    }
  }
  printf("\nBEFORE deletion :");
  traverse(start);

  if (start) {
    loc = start;
    start = start->next;
    printf("\nDELETED : %d, AT ADDRESS : %X", loc->data, loc);
    free(loc);
  } else {
    printf("Underflow!");
  }

  printf("\n\nAFTER Deletion :");
  traverse(start);
```

```
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : delete from firts postion of the linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ? 6
enter data for node 1 : 12
enter data for node 2 : 23
enter data for node 3 : 34
enter data for node 4 : 45
enter data for node 5 : 56
enter data for node 6 : 67

BEFORE deletion :
|12|A32928| -> |23|A32938| -> |34|A32948| -> |45|A32958| -> |56|A32968| -> |67|0|


DELETED : 12, AT ADDRESS : A32918

AFTER Deletion :
|23|A32938| -> |34|A32948| -> |45|A32958| -> |56|A32968| -> |67|0|
```

Program 12. WAP to delete a node at the end of the linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;
struct node {
  int data;
  node *next;
};

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) exit(1);
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
}
```

```
node *deleteFromEnd(node *start) {
  node *loc, *locp;
  loc = locp = start;

  while (loc->next) {
    locp = loc;
    loc = loc->next;
  }
  locp->next = NULL;
  printf("\nDELETED : %d, AT ADDRESS : %X\n", loc->data, loc);
  free(loc);
}
int main() {
  printIntro("delete from END of the linked list.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ? ");
  scanf("%d", &n);
  for (int data , i = 0; i < n; i++) {
    printf("enter data for node %d : ", i + 1);
    scanf("%d", &data);
    if (start == NULL) {
      loc = getNode(data);
      start = loc;
    } else {
      loc->next = getNode(data);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);
```

```
  if (start == NULL)
    printf("underflow!");          // no node exists
  else if (start->next == NULL) {  // only one node exists
    loc = start;
    printf("\nDELETED : %d, AT ADDRESS : %X\n", loc->data, loc);
    free(loc);
    start = NULL;
  } else
    deleteFromEnd(start);  // multiple node exists

  printf("\nafter Deletion : \n");
  traverse(start);

  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : delete from END of the linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ? 5
enter data for node 1 : 12
enter data for node 2 : 34
enter data for node 3 : 453
enter data for node 4 : 45
enter data for node 5 : 5555
nodes in linked list :

|12|32928| -> |34|32938| -> |453|32948| -> |45|32958| -> |5555|0|


DELETED : 5555, AT ADDRESS : 32958

after Deletion :

|12|32928| -> |34|32938| -> |453|32948| -> |45|0|
```

Program 13. WAP to delete a node at the specific position of the linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  while (loc) {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
```

```
}

node *deleteFromPosition(node *start, const int position) {
  int currentPosition = 0;
  node *loc, *locp;
  loc = locp = start;

  while (currentPosition < position) {
    locp = loc;
    loc = loc->next;
    currentPosition++;
  }
  locp->next = loc->next;
  printf("\nDELETED : %d, AT ADDRESS : %X\n", loc->data, loc);
  free(loc);
}
int main() {
  printIntro("delete from given postion of the linked list.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ? ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    if (start == NULL) {
      loc = getNode(1);
      start = loc;
    } else {
      loc->next = getNode((i + 1) * 3);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);
```

```
  int position;
  printf("enter position (indexing from 0) node to be deleted from : ");
  scanf("%d", &position);

  if (position < 0 || position >= n) {
    printf("invalid position!");
    return 1;
  }

  if (position == 0) {
    loc = start;
    start = start->next;
    printf("\nDELETED : %d, AT ADDRESS : %X\n", loc->data, loc);
    free(loc);
  } else {
    deleteFromPosition(start, position);
  }
  printf("\nafter Deletion : \n");
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : delete from given postion of the linked list.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ? 6
nodes in linked list :

|1|C52928| -> |6|C52938| -> |9|C52948| -> |12|C52958| -> |15|C52968| -> |18|0|

enter position (indexing from 0) node to be deleted from : 3

DELETED : 12, AT ADDRESS : C52948

after Deletion :

|1|C52928| -> |6|C52938| -> |9|C52958| -> |15|C52968| -> |18|0|
```

Program 14. WAP to create a grounded header linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

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

node *getNode(int data, node *start) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) exit(1);
  newNode->data = data;
  newNode->next = NULL;
  start->data += 1 ; // UPDATING VALUE AT HEADER
  return newNode;
}
void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  while (loc){
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  }
  printf("\n\n");
```

```
}
void insertAtLastNode(node *start, int data) {
  node *loc = start;
  // get last node
  while (loc->next) {
    loc = loc->next;
  }
  loc->next = getNode(data, start);
}

int main() {
  printIntro("LINKED LIST with header node.");
  node HEADER = {0,NULL};
  node *start = &HEADER;
  node *loc = &HEADER;

  int n;
  printf("how many nodes linked list should contain ?  : ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    if (start == NULL) {
      loc = getNode(1, start);
      start->next = loc;
    } else {
      loc->next = getNode((i + 1) * 3, start);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);

  int data;
  printf("enter item to be inserted at end : ");

  // creating and inserting node at END
```

```
  scanf("%d", &data);
  if (start)
    insertAtLastNode(start, data);
  else
    start->next = getNode(data, start);

  printf("After insertion: \n");
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : LINKED LIST with header node.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ?  : 5
nodes in linked list :

|5|682918| -> |3|682928| -> |6|682938| -> |9|682948| -> |12|682958| -> |15|0|

enter item to be inserted at end : 44444
After insertion:

|6|682918| -> |3|682928| -> |6|682938| -> |9|682948| -> |12|682958| -> |15|682968| -> |44444|0|
```

Program 15. WAP to create a circular linked list.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"
typedef struct node node;

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

node *getNode(int data, node *start) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) exit(1);
  newNode->data = data;
  newNode->next = start;
  return newNode;
}

void traverse(node *start) {
  node *loc = start;
  if (!loc) {
    printf("no node exist!\n");
    return;
  }

  printf("\n");
  do {
    printf("|%d|%X|", loc->data, loc->next);
    if (loc->next) printf(" -> ");
    loc = loc->next;
  } while (loc != start);
  printf("\n\n");
}
```

```
void insertAtLastNode(node *start, int data) {
  node *loc = start;
  // get last node
  do {
    loc = loc->next;
  } while (loc->next != start);
  loc->next = getNode(data, start);
}

int main() {
  printIntro("inserting end of the CIRCULAR LINKED LIST.");
  node *start = NULL;
  node *loc = NULL;

  int n;
  printf("how many nodes linked list should contain ?  : ");
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    if (start == NULL) {
      loc = getNode(1, start);
      start = loc;
    } else {
      loc->next = getNode((i + 1) * 3, start);
      loc = loc->next;
    }
  }
  printf("nodes in linked list : \n");
  traverse(start);

  int data;
  printf("enter item to be inserted at end : ");

  // creating and inserting node at END
  scanf("%d", &data);
  if (start)
```

```
    insertAtLastNode(start, data);
  else
    start = getNode(data, start);

  printf("After insertion: \n");
  traverse(start);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : inserting end of the CIRCULAR LINKED LIST.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
how many nodes linked list should contain ?  : 4
nodes in linked list :

|1|6D2928| -> |6|6D2938| -> |9|6D2948| -> |12|6D2918| ->

enter item to be inserted at end : 5555
After insertion:

|1|6D2928| -> |6|6D2938| -> |9|6D2948| -> |12|6D2958| -> |5555|6D2918| ->
```

Program 16. WAP to perform push operation in stack using array at compile time.

Code:

```
#include <stdio.h>
#include "intro.c"
#define MAX 5  // Defining the maximum size of the stack
int stack[MAX];  // Stack array declaration
int top = -1;    // Stack top initialization

// Function to return the top element from the stack
void peek() {
  if (top >= 0) {
    printf("%d\n", stack[top]);
    return;
  } else {
    printf("Stack is empty\n");
    return;
  }
}
// Function to add an element to the stack
void push(int item) {
  if (top < MAX - 1) {
    top++;
    stack[top] = item;
    printf("pushed %d\n", item);
    return;
  }
  printf("Stack Overflow\n");
}

// Function to remove the top element from the stack
void pop() {
  if (top >= 0) {
    printf("popped %d\n", stack[top]);
    top--;
```

```
    return;
  } else {
    printf("Stack Underflow\n");
    return;
  }
}

int main(int argc, char const *argv[]) {
  printIntro("push operation on array implementation of stack");
  push(5);
  push(6);
  push(8);
  push(9);
  push(2);
  push(3);

  printf("\n");
  pop();
  pop();
  pop();
  pop();
  pop();
  pop();
  return 0;
}
```

Output:

```
Author : Jitendra Kumar Sahu
Program topic : push operation on array implementation of stack
pushed 5
pushed 6
pushed 8
pushed 9
pushed 2
Stack Overflow

popped 2
popped 9
popped 8
popped 6
popped 5
Stack Underflow
```

Program 17. WAP to perform pop operation in stack using array at compile time.

Code:

```
#include <stdio.h>
#include "intro.c"
#define MAX 5  // Defining the maximum size of the stack

int stack[MAX];  // Stack array declaration
int top = -1;    // Stack top initialization

// Function to return the top element from the stack
void peek() {
  if (top >= 0) {
    printf("%d\n", stack[top]);
    return;
  } else {
    printf("Stack is empty\n");
    return;
  }
}
// Function to add an element to the stack
void push(int item) {
  if (top < MAX - 1) {
    top++;
    stack[top] = item;
    printf("pushed %d\n", item);
    return;
  }
  // else
  printf("Stack Overflow\n");
}

// Function to remove the top element from the stack
void pop() {
  if (top >= 0) {
    printf("popped %d\n", stack[top]);
```

```
    top--;
    return;
  } else {
    printf("Stack Underflow\n");
    return;
  }
}

int main(int argc, char const *argv[]) {
  printIntro("POP operation on array implementation of stack");
  push(5);
  push(6);
  push(8);
  push(9);
  push(2);
  push(3);

  printf("\n");
  pop();
  pop();
  pop();
  pop();
  pop();
  pop();
  return 0;
}
```

Output:

```
Author : Jitendra Kumar Sahu
Program topic : POP operation on array implementation of stack
pushed 12
pushed 16
pushed 18
pushed 19
pushed 21
Stack Overflow

popped 21
popped 19
popped 18
popped 16
popped 12
Stack Underflow
```

Program 18. Write AProgram to implement Stack operation in array using switch case.

Code:

```
#include <stdio.h>
#include "intro.c"
#define MAX 5  // Defining the maximum size of the stack

int stack[MAX];  // Stack array declaration
int top = -1;    // Stack top initialization

// Function to return the top element from the stack
void peek() {
  if (top >= 0) {
    printf("%d\n", stack[top]);
    return;
  } else {
    printf("Stack is empty\n");
    return;
  }
}
// Function to add an element to the stack
void push(int item) {
  if (top < MAX - 1) {
    top++;
    stack[top] = item;
    //printf("pushed %d\n", item);
    return;
  }
  // else
  printf("Stack Overflow\n");
}

// Function to remove the top element from the stack
void pop() {
  if (top >= 0) {
```

```
    printf("popped %d\n", stack[top]);
    top--;
    return;
  } else {
    printf("Stack Underflow\n");
    return;
  }
}

void traverse(){
if (top < 0) return ;
for(int i = top ; i >= 0 ; i--){
printf("%d ",stack[i]) ;
}
printf("\n") ;
}
int main() {
  printIntro(" implement Stack operation in array using switch case. ");
  printf("Enter\n1 to push\n");
  printf("2 to pop\n");
  printf("3 to print stack\n");
  printf("0 to exit!\n");
  while (1) {
    int choice;
    printf(">>");
    scanf("%d", &choice);
    int data;
    switch (choice) {
      case 1:  // code to call push on stack
        printf("enter data : ");
        scanf("%d", &data);
        push(data) ;
        break;
      case 2:  // code to call pop from stack
        pop();
```

```
        break;
      case 3:
        printf("printing stack\n");
        traverse();
        break;
      default:
        printf("exiting........");
        return 0;
    }
  }
  return 0;
}
```

Output:

```
Author : Jitendra Kumar Sahu
Program topic : implement Stack operation in array using switch case.
Enter
1 to push
2 to pop
3 to print stack
0 to exit!
>>1
enter data : 45
>>1
enter data : 67
>>1
enter data : 98
>>1
enter data : 90
>>1
enter data : 56
>>1
enter data : 35
Stack Overflow
>>3
printing stack
56 90 98 67 45
>>2
popped 56
>>2
popped 90
>>
3
printing stack
98 67 45
>>1
enter data : 1111
>>3
printing stack
1111 98 67 45
>>0
exiting........
```

Program 19. WAP to perform push operation in stack using linked list.

Code:

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

#include "intro.c"
typedef struct node node;
struct node {
  int data;
  node *next;
};

node *createNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) {
    printf("overflowed!");
    exit(1);
  }
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}

void traverse(node *start) {
  while (start) {
    printf("%d ", start->data);
    start = start->next;
  }
}

int main() {
  printIntro("linked list implementation of stack");
  node *top = NULL;
  printf("Enter\n1 to push\n");
  printf("2 to pop\n");
```

```
  printf("3 to print stack\n");
  printf("0 to exit!");
  while (1) {
    int choice;
    printf("\n>>");
    scanf("%d", &choice);
    int data;
    switch (choice) {
      case 1:  // code to push onto stack
        printf("enter data : ");
        scanf("%d", &data);
        node *loc = createNode(data);
        if (top) {
          loc->next = top;
          top = loc;
        } else {
          top = loc;
        }
        break;
      case 2:  // code to pop from stack
        if (top) {
          printf("popped : %d\n", top->data);
          top = top->next;
        } else {
          printf("underflow!\n");
        }
        break;
      case 3:
        printf("printing stack\n");
        traverse(top);
        break;
      default:
        printf("exiting........");
        return 0;
    }
```

```
  }
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : linked list implementation of stack

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter
1 to push
2 to pop
3 to print stack
0 to exit!
>>1
enter data : 23

>>1
enter data : 45

>>1
enter data : 56

>>1
enter data : 78

>>1
enter data : 90

>>2
popped : 90

>>2
popped : 78

>>3
printing stack
56 45 23
>>1
enter data : 98

>>1
enter data : 65

>>3
printing stack
65 98 56 45 23
>>0
exiting.....
```

```
Author : Jitendra Kumar SAHU
Program Topic : linked list implementation of stack

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter
1 to push
2 to pop
3 to print stack
0 to exit!
>>1
enter data : 23

>>1
enter data : 45

>>1
enter data : 56

>>1
enter data : 78

>>1
enter data : 90

>>2
popped : 90

>>2
popped : 78

>>3
printing stack
56 45 23
>>1
enter data : 98

>>1
enter data : 65

>>3
printing stack
65 98 56 45 23
>>0
exiting........
```

Program 20. WAP to perform pop operation in stack using linked list.

Code:

```
#include <stdio.h>
 #include <stdlib.h>
 #include "intro.c"
 typedef struct node node;
 struct node {
   int data;
   node *next;
 };
 node *createNode(int data) {
   node *newNode = (node *)malloc(sizeof(node));
   if (newNode == NULL) {

    printf("overflowed!");
     exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
 }

void traverse(node *start) {
  while (start) {
    printf("%d ", start->data);
    start = start->next;
  }
}

int main() {

  printIntro("linked list implementation of stack");
    node *top = NULL;
  printf("Enter\n1 to push\n");
  printf("2 to pop\n");
```

```
  printf("3 to print stack\n");
  printf("0 to exit!\n");

  while (1) {
     int choice;
     printf(">>");
     scanf("%d", &choice);
     int data;
     switch (choice) {
       case 1:  // code to push onto stack
         printf("enter data : ");
         scanf("%d", &data);
         node *loc = createNode(data);
         if (top) {
          loc->next = top;
           top = loc;
         } else {
          top = loc;
        }
        break;
      case 2:  // code to pop from stack
        if (top) {
          printf("poped : %d\n", top->data);
          top = top->next;
        } else {
          printf("underflow!\n");
        }
        break;
      case 3:
        printf("printing stack\n");
        traverse(top);
        break;
      default:
        printf("exiting........");
```

```
        return 0;
    }
  }
  return 0;

}
```

Output:

```
Author : Jitendra Kumar Sahu
Program topic : linked list implementation of stack
Enter
1 to push
2 to pop
3 to print stack
0 to exit!
>>1
enter data : 23
>>1
enter data : 67
>>1
enter data : 98
>>1
enter data : 54
>>3
printing stack
54 98 67 23 >>1
enter data : 34
>>1
enter data : 90
>>3
printing stack
90 34 54 98 67 23 >>2
poped : 90
>>2
poped : 34
>>3
printing stack
54 98 67 23 >>0
exiting........
```

Program 21. WAP to find factorial of a given number using Recursion.

Code:

```
#include <stdio.h>
#include "intro.c"

long long fact(long long n) {
  if (n == 0 || n == 1) return 1;
  return n * fact(n - 1);
}

int main() {
  printIntro("Factorial using recursion");
  int num ;
  printf("Enter number to calculate factorial : " ) ;
  scanf("%d",&num) ;
  printf("factorial %d = %lld\n",num,fact(num)) ;
  return 0;
}
```

Output:

```
C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat factorial.c
Author : Jitendra Kumar SAHU
Program Topic : Factorial using recursion

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter number to calculate factorial : 5
factorial 5 = 120

C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat factorial.c
Author : Jitendra Kumar SAHU
Program Topic : Factorial using recursion

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter number to calculate factorial : 6
factorial 6 = 720
```

Program 22. WAP to find fibonacci series using Recursion.

Code:

```
#include <stdio.h>
#include "intro.c"

int fib(int n) {
  if (n == 0 || n == 1) return n;
  return fib(n - 1) + fib(n - 2);
}

int main() {
  printIntro("Topic");

  int n;
  printf("Enter lenght of fibonacci series : ");
  scanf("%d",&n) ;
  printf("Fibonacci series of %d terms : \n", n);
  for (int i = 0; i < n; i++) printf("%d ", fib(i));

  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : Topic

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter lenght of fibonacci series : 10
Fibonacci series of 10 terms :
0 1 1 2 3 5 8 13 21 34
```

Program 23. WAP to implement "Towers of Hanoi" problem using Recursion.

Code:

```
#include <stdio.h>
#include "intro.c"

void tower(int n , char from , char to , char aux){
   if (n == 1){
    printf("%c -> %c\n",from , to) ;
    return;
   }
   tower(n-1,from,aux,to);
   printf("%c -> %c\n",from , to) ;
   tower(n-1,aux,to,from) ;
}
 int main(int argc, char const *argv[])
{
    printIntro("Tower of hanoi") ;
    int n ;
    tower(3,'A','B','Z') ;
    return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : Tower of hanoi

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
A -> B
A -> Z
B -> Z
A -> B
Z -> A
Z -> B
A -> B
```

Program 24. WAP to insert an element into a QUEUE using array.

Code:

```
#include <stdio.h>
#include "intro.c"
#define MAX 5

typedef enum { false, true } boolean;

int FRONT = -1;
int REAR = -1;
int QUEUE[MAX];

boolean isOverflow() {
  if ((FRONT == 0 && REAR == MAX - 1) || FRONT == REAR + 1) return true;
  return false;
}

boolean isUnderflow() {
  if (FRONT == -1) return true;
  return false;
}

void insert(const int item) {
  if (isOverflow()) {
    printf("Overflow!\n");
    return;
  }
  // set rear's value
  if (FRONT == -1) {
    FRONT = REAR = 0;
  } else if (REAR == MAX - 1)
    REAR = 0;
  else
    REAR += 1;
  // set item
```

```
  QUEUE[REAR] = item;
}

void delete () {
  if (isUnderflow()) {
    printf("Underflow!\n");
    return;
  }
  // print item
  printf("deleted : %d \n", QUEUE[FRONT]);

  // set FRONT's position
  if (FRONT == REAR) {
    FRONT = REAR = -1;
  } else if (FRONT == MAX) {
    FRONT = 0;
  } else
    FRONT += 1;
}

int main() {
  printIntro("Array implementation of Circler Queue");
  printf(
      "Queue created;\n1 to insert item\n2 to delete from queue\n0 to "
      "exit out of program:\n");
  int s;
  while (true) {
    printf("enter option : ");
    scanf("%d", &s);
    switch (s) {
      case 0:
        return 0;
        break;
      case 1:
        printf("enter item to be inserted : ");
```

```
        int data;
        scanf("%d", &data);
        insert(data);
        break;
      case 2:
        delete ();
           break ;
      default:
        printf("enter valid number!\n");
        break;
    }
  }

  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : Array implementation of Circler Queue

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Queue created;
1 to insert item
2 to delete from queue
0 to exit out of program:
enter option : 1
enter item to be inserted : 21
enter option : 1
enter item to be inserted : 12
enter option : 1
enter item to be inserted : 23
enter option : 1
enter item to be inserted : 34
enter option : 1
enter item to be inserted : 45
enter option : 1
enter item to be inserted : 56
Overflow!
enter option : 1
enter item to be inserted : 455
Overflow!
enter option : 2
deleted : 21
enter option : 2
deleted : 12
enter option : 1
enter item to be inserted : 34
enter option : 1
enter item to be inserted : 78
enter option : 1
enter item to be inserted : 89
Overflow!
enter option : 2
deleted : 23
enter option : 2
deleted : 34
enter option : 2
deleted : 45
enter option : 2
deleted : 0
enter option : 2
deleted : 34
enter option : 2
deleted : 78
enter option : 2
Underflow!
```

Program 25. WAP to delete an element from a QUEUE using array.

Code:

```
#include <stdio.h>
#include "intro.c"
#define MAX 5
typedef enum { false, true } boolean;

int FRONT = -1;
int REAR = -1;
int QUEUE[MAX];

boolean isOverflow() {
  if ((FRONT == 0 && REAR == MAX - 1) || FRONT == REAR + 1) return true;
  return false;
}
boolean isUnderflow() {
  if (FRONT == -1) return true;
  return false;
}

void insert(const int item) {
  if (isOverflow()) {
    printf("Overflow!\n");
    return;
  }
  // set rear's value
  if (FRONT == -1) {
    FRONT = REAR = 0;
  } else if (REAR == MAX - 1)
    REAR = 0;
  else
    REAR += 1;
  // set item
  QUEUE[REAR] = item;
}
```

```
void delete () {
  if (isUnderflow()) {
    printf("Underflow!\n");
    return;
  }
  // print item
  printf("deleted : %d \n", QUEUE[FRONT]);

  // set FRONT's position
  if (FRONT == REAR) {
    FRONT = REAR = -1;
  } else if (FRONT == MAX) {
    FRONT = 0;
  } else
    FRONT += 1;
}

int main() {
  printIntro("Array implementation of Circler Queue");
  printf(
      "Queue created;\n1 to insert item\n2 to delete from queue\n0 to "
      "exit out of program:\n");
  int s;
  while (true) {
    printf("enter option : ");
    scanf("%d", &s);
    switch (s) {
      case 0:
        return 0;
        break;
      case 1:
        printf("enter item to be inserted : ");
        int data;
        scanf("%d", &data);
```

```
        insert(data);
        break;
      case 2:
        delete ();
           break ;
      default:
        printf("enter valid number!\n");
        break;
    }
  }

  return 0;
}
```

Output :

```
Author : Jitendra Kumar SAHU
Program Topic : Array implementation of Circler Queue

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Queue created;
1 to insert item
2 to delete from queue
0 to exit out of program:
enter option : 1
enter item to be inserted : 12
enter option : 1
enter item to be inserted : 23
enter option : 1
enter item to be inserted : 32
enter option : 1
enter item to be inserted : 45
enter option : 2
deleted : 12
enter option : 2
deleted : 23
enter option : 2
deleted : 32
enter option : 2
deleted : 45
enter option : 2
Underflow!
enter option : 0
```

Program 26. WAP to implement binary tree and perform preorder traversal using recursion/stack.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"

typedef struct node node;

struct node {
  int data;
  node *left;
  node *right;
};

node *createNode(int data) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) exit(1);
  newNode->data = data;
  newNode->left = newNode->right = NULL;
}

void preOrder(node *root) {
  if (root == NULL) return;
  printf("%d ", root->data);
  preOrder(root->left);
  preOrder(root->right);
}

int main() {
  printIntro("binary tree and pre-order traversal using recursion");
  node *root = NULL;
  node *n1 = createNode(1);
  node *n2 = createNode(5);
  node *n3 = createNode(6);
```

```
    node *n4 = createNode(45);
    node *n5 = createNode(67);
    node *n6 = createNode(66);
    node *n7 = createNode(81);

    root = n1;

    root->left = n2;
    root->right = n3;

    n2->left = n4;
    n2->right = n5;

    n3->left = n6;
    n3->right = n7;

    printf("pre-order traversal of binary tree : ");
    preOrder(root);
    return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : binary tree and pre-order traversal using recursion

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
pre-order traversal of binary tree : 1 5 45 67 6 66 81
```

Program 27. WAP to implement binary tree and perform inorder traversal using recursion/stack.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"
typedef struct node node;

struct node {
  int info;
  node *left;
  node *right;
};

node *createNode(int info) {
  node *newNode = (node *)malloc(sizeof(node));
  if (newNode == NULL) exit(1);
  newNode->info = info;
  newNode->left = newNode->right = NULL;
}

void inorder(node *root) {
  if (root == NULL) return;
  inorder(root->left);
  printf("%d ", root->info);
  inorder(root->right);
}

int main() {
  printIntro("binary tree and  in-order traversal using recursion");
  node *n1 = createNode(1);
  node *n2 = createNode(5);
  node *n3 = createNode(6);
  node *n4 = createNode(45);
  node *n5 = createNode(67);
```

```
  node *n6 = createNode(66);
  node *n7 = createNode(81);

  node *root = n1;

  root->left = n2;
  root->right = n3;

  n2->left = n4;
  n2->right = n5;

  n3->left = n6;
  n3->right = n7;

  printf("in-order traversal of binary tree : ");
  inorder(root);
  printf("\n");
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : binary tree and  in-order traversal using recursion

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
in-order traversal of binary tree : 45 5 67 1 66 6 81
```

Program 28. WAP to implement binary tree and perform postorder traversal using recursion/stack.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include "intro.c"
typedef struct node node;

struct node {
  int info;
  node *left;
  node *right;
};

node *createNode(int info) {
  node *newNode = (node *)malloc(sizeof(node));
  if(newNode==NULL) exit(1) ;
  newNode->info = info;
  newNode->left = newNode->right = NULL;
}

void postOrder(node *root) {
  if (root == NULL) return;
  postOrder(root->left);
  postOrder(root->right);
  printf("%d ", root->info);
}

int main() {
  printIntro("binary tree and post-order traversal using recursion");
   node *root = NULL;
  node *n1 = createNode(1);
  node *n2 = createNode(5);
  node *n3 = createNode(6);
  node *n4 = createNode(45);
```

```
  node *n5 = createNode(67);
  node *n6 = createNode(66);
  node *n7 = createNode(81);

  root = n1;

  root->left = n2;
  root->right = n3;

  n2->left = n4;
  n2->right = n5;

  n3->left = n6;
  n3->right = n7;

  printf("post order traversal of binary tree : ") ;
  postOrder(root);
  printf("\n");
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : binary tree and post-order traversal using recursion

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
post order traversal of binary tree : 45 67 5 66 81 6 1
```

Program 29. WAP to implement Linear search algorithm.

Code:

```
#include <stdio.h>
 #include "intro.c"
 void search(int key, int arr[], int l) {
  for (int i = 0; i < l; i++) {
    if (key == arr[i]) {
      printf("found %d at %d index\n", key, i);
      return;
    }
  }
  printf("element not found\n");
}

int main() {
  printIntro("Topic");

  int key, n;
  printf("enter number of element in array : ");
  scanf("%d", &n);
  int arr[n];
  printf("enter elements : \n");

  for (int i = 0; i < n; i++) {
    printf("enter element %d : ", (i + 1));
    scanf("%d", &arr[i]);
  }
  printf("all items are inserted!\n");
  printf("enter item to be searched : ");
  scanf("%d", &key);
  search(key, arr, n);
  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : Linear search

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter number of element in array : 6
enter elements :
12 33 45 65 78 98
all items are inserted!
enter item to be searched : 78
found 78 at 4 index

C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat linearSearch.c
Author : Jitendra Kumar SAHU
Program Topic : Linear search

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter number of element in array : 6
enter elements :
12 33 45 65 78 98
all items are inserted!
enter item to be searched : 9999
element not found
```

Program 30. WAP to implement Binary search algorithm.

Code:

```
#include <stdio.h>

#include "intro.c"
// *** there is problem in binary search algo
// larger item which are not in the list are shown to be at
// at 9th index YOU HAVE to FIX it
int search(int *arr, int n, int key) {
  int start = 0, end = n, mid;
  while (start <= end) {
    mid = (start + end) / 2;
    int currentValue = arr[mid];
    if (currentValue == key)
      return mid;
    else if (currentValue < key)
      start = mid + 1;
    else
      end = mid - 1;
  }
  return -1;
}

int main() {
  printIntro("Linear search");

  int key;
  const int n = 9;
  int arr[] = {11, 12, 13, 14, 15, 16, 17, 18, 19};
  printf("The array is : ");
  for (size_t i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }

  printf("enter item to be searched : ");
```

```
  scanf("%d", &key);
  int res = search(arr, n-1, key);
  if (res == -1) {
    printf("element not found\n");
  } else {
    printf("element %d found at index %d\n", key, res);
  }
  return 0;
}

// int searchRecursivly(int *arr, int key, int start , int end){
//    if (start >= end) return -1 ;
//    int mid = ( start + end ) / 2 ;
//    int currVlu = arr[mid] ;
//    if (currVlu == key) return mid ;
//    else if(currVlu < key )return searchRecursivly(arr,key,start+1,end) ;
//    else return searchRecursivly(arr,key,start,end-1) ;
// }
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : Linear search

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The array is : 11 12 13 14 15 16 17 18 19 enter item to be searched : 15
element 15 found at index 4

C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat binarySearch.c
Author : Jitendra Kumar SAHU
Program Topic : Linear search

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The array is : 11 12 13 14 15 16 17 18 19 enter item to be searched : 17
element 17 found at index 6

C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat binarySearch.c
Author : Jitendra Kumar SAHU
Program Topic : Linear search

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The array is : 11 12 13 14 15 16 17 18 19 enter item to be searched : -78
element not found

C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat binarySearch.c
Author : Jitendra Kumar SAHU
Program Topic : Linear search

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The array is : 11 12 13 14 15 16 17 18 19 enter item to be searched : 20
element not found
```

Program 31. WAP to implement Bubble sort.

Code:

```
#include <stdio.h>
#include "intro.c"

void bubbleSort(int arr[], int n) {
  for (size_t i = 0; i < n - 1; i++) {
    for (size_t j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        // swap item
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}
int main() {
  printIntro("Bubble sort implementation");

  int n;
  printf("enter the number of items in array : ");
  scanf("%d", &n);
  int arr[n];

  printf("enter items of array : ");
  for (size_t i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
  bubbleSort(arr, n);
  for (size_t i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
```

```
  return 0;
}
```

Output:

```
C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assi
Sort.c
Author : Jitendra Kumar SAHU
Program Topic : Bubble sort implementation

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter the number of items in array : 6
enter items of array : 99 88 77 555 41 36
36 41 77 88 99 555
```

Program 32. WAP to implement Insertion sort.

Code:

```
#include <stdio.h>

#include "intro.c"

// Insertion Sort
void insertionSort(int arr[], int n) {
  int i, key, j;
  for (i = 1; i < n; i++) {
    key = arr[i];
    j = i - 1;
    while (j >= 0 && arr[j] > key) {
      arr[j + 1] = arr[j];
      j = j - 1;
    }
    arr[j + 1] = key;
  }
}

int main() {
  printIntro("implementation of Insertion Sort");

  int n;
  printf("enter the number of items in array : ");
  scanf("%d", &n);
  int arr[n];
   printf("enter items of array : ");
  for (size_t i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
  insertionSort(arr, n);

  printf("Sorted array: ");
  for (int i = 0; i < n; i++) printf("%d ", arr[i]);
```

```
  printf("\n");


  return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : implementation of Insertion Sort

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter the number of items in array : 7
enter items of array : 12 56 23 67 98 54 3
Sorted array: 3 12 23 54 56 67 98
```

Program 33. WAP to implement Selection sort.

Code:

```
#include <stdio.h>
#include "intro.c"

// Selection Sort
void selectionSort(int arr[], int n) {
    int i, j, min_idx;
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;
        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    printIntro("implementation of Selection Sort");
 int n;
  printf("enter the number of items in array : ");
  scanf("%d", &n);
  int arr[n];

  printf("enter items of array : ");
  for (size_t i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
    selectionSort(arr, n);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
```

```
    printf("\n");


    return 0;
}
```

Output:

```
C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignment\programs>execC.bat selectionSort.c
Author : Jitendra Kumar SAHU
Program Topic : implementation of Selection Sort

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter the number of items in array : 4
enter items of array : 888 777 333 111
Sorted array: 111 333 777 888
```

Program 34. WAP to implement Merge sort.

Code:

```
#include <stdio.h>
#include "intro.c"

// Merge Sort
void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
```

```
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int main() {
    printIntro("implementation of Merge Sort");

 int n;
  printf("enter the number of items in array : ");
  scanf("%d", &n);
  int arr[n];

  printf("enter items of array : ");
  for (size_t i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
    mergeSort(arr, 0, n - 1);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

Output:

```
Author : Jitendra Kumar SAHU
Program Topic : implementation of Merge Sort

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter the number of items in array : 7
enter items of array : 87 56 4 34 2 5 7
Sorted array: 2 4 5 7 34 56 87
```

Program 35. WAP to implement Quick sort.

Code:

```
 #include <stdio.h>

#include "intro.c"

// Quick Sort
int partition(int arr[], int low, int high) {
  int pivot = arr[high];
  int i = (low - 1);
  for (int j = low; j <= high - 1; j++) {
    if (arr[j] < pivot) {
      i++;
      int temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
    }
  }
  int temp = arr[i + 1];
  arr[i + 1] = arr[high];
  arr[high] = temp;
  return (i + 1);
}

void quickSort(int arr[], int low, int high) {
  if (low < high) {
    int pi = partition(arr, low, high);
    quickSort(arr, low, pi - 1);
    quickSort(arr, pi + 1, high);
  }
}

int main() {
  printIntro("implementation of Quick Sort");
```

```
  int n;
  printf("enter the number of items in array : ");
  scanf("%d", &n);
  int arr[n];

  printf("enter items of array : ");
  for (size_t i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
  quickSort(arr, 0, n - 1);

  printf("Sorted array: ");
  for (int i = 0; i < n; i++) printf("%d ", arr[i]);
  printf("\n");

  return 0;
}
```

Output:

```
C:\Users\Jitendra Sahu GT\Nextcloud\MCA\DSA\assignmen
Author : Jitendra Kumar SAHU
Program Topic : implementation of Quick Sort

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
enter the number of items in array : 8
enter items of array : 12 89 45 32 78 90 4 -23
Sorted array: -23 4 12 32 45 78 89 90
```