Nidhi Bisen
0801IT243D03 (B3-BATCH)

LAB ASSIGNMENT-1

Q1) write a program to print largest number among the given number in array using dynamic array.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, largest;
    printf("Enter the value of n\n");
    scanf("%d", &n);
    int *A;
    A = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
    {
        printf("enter the value of element %d:- \n", i + 1);
        scanf("%d", &A[i]);
    }
    largest = A[0];
    for (int i = 0; i < n; i++)
    {
        if (largest < A[i])
        {
            largest = A[i];
        }
    }
    printf("largest no:- %d", largest);
    printf("\n Nidhi Bisen Assignment Q-1");
}
```

Output:

```
PS C:\Users\User\OneDrive\Desktop\DSA> gcc first.c
PS C:\Users\User\OneDrive\Desktop\DSA> ./a
Enter the value of n
3
enter the value of element 1:-
23
enter the value of element 2:-
44
enter the value of element 3:-
2
largest no:- 44
 Nidhi Bisen Assignment Q-1
PS C:\Users\User\OneDrive\Desktop\DSA>
```

Nidhi Bisen
0801IT243D03 (B3-BATCH)

Q2) write a program to print the smallest number among the given number in array using dynamic array.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, small;
    printf("Enter the value of n\n");
    scanf("%d", &n);
    int *A;
    A = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
    {
        printf("enter the value of element %d:- \n", i + 1);
        scanf("%d", &A[i]);
    }
    small = A[0];
    for (int i = 0; i < n; i++)
    {
        if (small > A[i])
        {
            small = A[i];
        }
    }
    printf("small no:- %d", small);
    printf("\n Nidhi Bisen Assignment Q-2");
}
```

OUTPUT :-

```
PS C:\Users\User\OneDrive\Desktop\DSA> ./a
Enter the value of n
4
enter the value of element 1:-
33
enter the value of element 2:-
2
enter the value of element 3:-
1
enter the value of element 4:-
5
small no:- 1
 Nidhi Bisen Assignment Q-2
PS C:\Users\User\OneDrive\Desktop\DSA>
```

Nidhi Bisen
0801IT243D03 (B3-BATCH)

Q3) write a program to print the elements of Array in reverse order show the use of malloc and free function in your program .

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int initialSize = 5;
    int newSize = 10;
    int i;
    int *array = (int *)malloc(initialSize * sizeof(int));
    if (array == NULL)
    {
        printf("Memory allocation failed!\n");
        return 1;
    }
    for (i = 0; i < initialSize; i++)
    {
        array[i] = i + 1;
    }
    array = (int *)realloc(array, newSize * sizeof(int));
    if (array == NULL)
    {
        printf("Memory reallocation failed!\n");
        return 1;
    }
    for (i = initialSize; i < newSize; i++)
    {
        array[i] = i + 1;
    }
    printf("Array elements in reverse order:\n");
    for (i = newSize - 1; i >= 0; i--)
    {
        printf("%d ", array[i]);
    }
    printf("\n");
    printf("\n Nidhi Bisen Assignment Q-3");
    free(array);
    return 0;
}
```

Nidhi Bisen
0801IT243D03 (B3-BATCH)

OUTPUT

```
PS C:\Users\User\OneDrive\Desktop\DSA> gcc first.c
PS C:\Users\User\OneDrive\Desktop\DSA> ./a
Array elements in reverse order:
10 9 8 7 6 5 4 3 2 1

 Nidhi Bisen Assignment Q-3
PS C:\Users\User\OneDrive\Desktop\DSA>
```

Nidhi Bisen
0801IT243D03 (B3-BATCH)

Q4) write a program in C to find the occurrence of duplicate element in array .

```
#include <stdio.h>
#include <stdlib.h>
void findDuplicates(int *arr, int size)
{
    int i, j;
    int *counts = (int *)malloc(size * sizeof(int));
    if (counts == NULL)
    {
        printf("Memory allocation failed for counts array!\n");
        return;
    }
    for (i = 0; i < size; i++)
    {
        counts[i] = 0;
    }
    for (i = 0; i < size; i++)
    {
        if (counts[i] == 0)
        {
            int count = 1;
            for (j = i + 1; j < size; j++)
            {
                if (arr[i] == arr[j])
                {
                    count++;
                    counts[j] = -1;
                }
            }
            if (count > 1)
            {
                printf("Element %d occurs %d times\n", arr[i],
                       count);
            }
        }
    }
    free(counts);
}
int main()
{
    int size;
    int *array;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &size);
    array = (int *)malloc(size * sizeof(int));
    if (array == NULL)
    {
```

```
        printf("Memory allocation failed!\n");
        return 1;
    }
    printf("Enter %d elements:\n", size);
    for (int i = 0; i < size; i++)
    {
        scanf("%d", &array[i]);
    }
    findDuplicates(array, size);
    printf("\n Nidhi Bisen Assignment Q-4");
    free(array);
    return 0;
}
```

Output:

```
PS C:\Users\User\OneDrive\Desktop\DSA> ./a
Enter the number of elements in the array: 4
Enter 4 elements:
1
4
5
4
Element 4 occurs 2 times

 Nidhi Bisen Assignment Q-4
PS C:\Users\User\OneDrive\Desktop\DSA> S
```

Q5) Write a menu driven program to implement following operations on the singly linked list. (a) Insert a node at the front of the linked list. (b) Insert a node at the end of the linked list. (c) Insert a node such that linked list is in ascending order.(according to info. Field) (d) Delete a first node of the linked list. (e) Delete a node before specified position. (f) Delete a node after specified position.

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

struct node {
    int data;
    struct node *next;
} *start = NULL;

void make() {
    printf("You are in make function\n");
    int g;
    struct node *newnode, *currentnode;
    while (1) {
        newnode = (struct node *)malloc(sizeof(struct node)); //
dynamic memory allocation
        printf("Enter the data :- \n");
        scanf("%d", &newnode->data); // input data
        newnode->next = NULL;
        if (start == NULL) {
            start = newnode;
            currentnode = newnode;
        } else {
            currentnode->next = newnode;
            currentnode = newnode;
        }
        printf("Press 1 to add a new node or 2 for exit: ");
        scanf("%d", &g);

        if (g == 2) {
            break;
        }
    }
}

void show() {
    printf("You are in show function\n");
    struct node *ptr;
    ptr = start;
    while (ptr != NULL) {
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
```

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

void beforestart() {
    printf("You are in beforestart function\n");
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node)); // dynamic
memory allocation
    printf("Enter the data :- \n");
    scanf("%d", &newnode->data); // input data
    newnode->next = start;
    start = newnode;
}

void afterlast() {
    printf("You are in afterlast function\n");
    struct node *newnode, *ptr;
    newnode = (struct node *)malloc(sizeof(struct node)); // dynamic
memory allocation
    printf("Enter the data :- \n");
    scanf("%d", &newnode->data); // input data
    newnode->next = NULL;

    if (start == NULL) {
        start = newnode;
    } else {
        ptr = start;
        while (ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = newnode;
    }
}

void ascen() {
    printf("You are in ascending order insertion function\n");
    struct node *newnode, *pre = NULL, *ptr;
    newnode = (struct node *)malloc(sizeof(struct node)); // dynamic
memory allocation
    printf("Enter the data :- \n");
    scanf("%d", &newnode->data); // input data
    newnode->next = NULL;

    if (start == NULL || newnode->data < start->data) {
        newnode->next = start;
        start = newnode;
    } else {
        ptr = start;
        while (ptr != NULL && newnode->data > ptr->data) {
            pre = ptr;
```

```
            ptr = ptr->next;
        }
        newnode->next = pre->next;
        pre->next = newnode;
    }
}

void delfn() {
    printf("You are in delete first node function\n");
    struct node *temp;
    if (start == NULL) {
        printf("There is nothing to delete\n");
    } else {
        temp = start;
        start = start->next;
        free(temp);
        printf("First node deleted successfully\n");
    }
}

void delbs() {
    printf("You are in delete before specified position
function\n");
    struct node *pre = NULL, *ptr, *pst;
    int value;
    printf("Enter the data of the node before which you want to
delete:\n");
    scanf("%d", &value);

    if (start == NULL || start->next == NULL) {
        printf("Not enough nodes to perform operation\n");
        return;
    }

    ptr = start;
    while (ptr->next != NULL && ptr->next->data != value) {
        pre = ptr;
        ptr = ptr->next;
    }

    if (ptr->next == NULL) {
        printf("Specified node not found\n");
    } else {
        pst = pre->next;
        pre->next = ptr->next;
        free(ptr);
        printf("Node before specified position deleted\n");
    }
}
```

```
void delas() {
    printf("You are in delete after specified position function\n");
    struct node *ptr, *pst;
    int value;
    printf("Enter the data of the node after which you want to
delete:\n");
    scanf("%d", &value);

    ptr = start;
    while (ptr != NULL && ptr->data != value) {
        ptr = ptr->next;
    }

    if (ptr == NULL || ptr->next == NULL) {
        printf("Node not found or there is no node after the
specified node\n");
    } else {
        pst = ptr->next;
        ptr->next = pst->next;
        free(pst);
        printf("Node after specified position deleted\n");
    }
}

int main() {
    char ch;
    while (1) {
        printf("Press (m) to: Make the linked list.\n");
        printf("Press (s) to: Show the linked list.\n");
        printf("Press (a) to: Insert a node at the front.\n");
        printf("Press (b) to: Insert a node at the end.\n");
        printf("Press (c) to: Insert a node in ascending order.\n");
        printf("Press (d) to: Delete the first node.\n");
        printf("Press (e) to: Delete a node before specified
position.\n");
        printf("Press (f) to: Delete a node after specified
position.\n");
        printf("Press (q) to: Exit.\n");
        scanf(" %c", &ch);

        switch (ch) {
            case 'm':
                make();
                break;
            case 's':
                show();
                break;
            case 'a':
                beforestart();
                break;
```

```
            case 'b':
                afterlast();
                break;
            case 'c':
                ascen();
                break;
            case 'd':
                delfn();
                break;
            case 'e':
                delbs();
                break;
            case 'f':
                delas();
                break;
            case 'q':
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
```

}

Output:

Nidhi Bisen
0801IT243D03 (B3-BATCH)

```
PS C:\Users\User\OneDrive\Desktop\DSA> gcc first.c
PS C:\Users\User\OneDrive\Desktop\DSA> ./a
Press (m) to: Make the linked list.
Press (s) to: Show the linked list.
Press (a) to: Insert a node at the front.
Press (b) to: Insert a node at the end.
Press (c) to: Insert a node in ascending order.
Press (d) to: Delete the first node.
Press (e) to: Delete a node before specified position.
Press (f) to: Delete a node after specified position.
Press (q) to: Exit.
m
You are in make function
Enter the data :-
33
Press 1 to add a new node or 2 for exit: 6
Enter the data :-
67
Press 1 to add a new node or 2 for exit: 1
Enter the data :-
33
Press 1 to add a new node or 2 for exit:
1
Enter the data :-
2
Press 1 to add a new node or 2 for exit:
1
Enter the data :-
44
Press 1 to add a new node or 2 for exit: 1
Enter the data :-
33
Press 1 to add a new node or 2 for exit: 1
Enter the data :-
34
Press 1 to add a new node or 2 for exit:
2
```

```
Press (f) to: Delete a node after specified position.
Press (q) to: Exit.
s
You are in show function
33 -> 67 -> 33 -> 2 -> 44 -> 33 -> 34 -> NULL
Press (m) to: Make the linked list.
Press (s) to: Show the linked list.
Press (a) to: Insert a node at the front.
Press (b) to: Insert a node at the end.
Press (c) to: Insert a node in ascending order.
Press (d) to: Delete the first node.
Press (e) to: Delete a node before specified position.
Press (f) to: Delete a node after specified position.
Press (q) to: Exit.
c
You are in ascending order insertion function
Enter the data :-
2
Press (m) to: Make the linked list.
Press (s) to: Show the linked list.
Press (a) to: Insert a node at the front.
Press (b) to: Insert a node at the end.
Press (c) to: Insert a node in ascending order.
Press (d) to: Delete the first node.
Press (e) to: Delete a node before specified position.
Press (f) to: Delete a node after specified position.
Press (q) to: Exit.
s
You are in show function
2 -> 33 -> 67 -> 33 -> 2 -> 44 -> 33 -> 34 -> NULL
Press (m) to: Make the linked list.
Press (s) to: Show the linked list.
Press (a) to: Insert a node at the front.
Press (b) to: Insert a node at the end.
Press (c) to: Insert a node in ascending order.
Press (d) to: Delete the first node.
Press (e) to: Delete a node before specified position.
```

```
Press (q) to: Exit.
s
You are in show function
2 -> 33 -> 67 -> 33 -> 2 -> 44 -> 33 -> 34 -> NULL
Press (m) to: Make the linked list.
Press (s) to: Show the linked list.
Press (a) to: Insert a node at the front.
Press (b) to: Insert a node at the end.
Press (c) to: Insert a node in ascending order.
Press (d) to: Delete the first node.
Press (e) to: Delete a node before specified position.
Press (f) to: Delete a node after specified position.
Press (q) to: Exit.
```

6) Write a menu driven program to implement following operations on the doubly linked list. (a) Insert a node at the front of the linked list. (b) Insert a node at the end of the linked list. (c) Insert a node such that linked list is in ascending order.(according to info. Field) (d) Delete a first node of the linked list. (e) Delete a node before specified position. (f) Delete a node after specified position.

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

struct node {
 int data;
   struct node *next;
   struct node *prev;
} *start = NULL;

void make() {
   int g;
   struct node *newnode, *currentnode;
   while (1) {
      newnode = (struct node *)malloc(sizeof(struct node));
      printf("Enter the data: \n");
      scanf("%d", &newnode->data);
      newnode->next = NULL;
      newnode->prev = NULL;

      if (start == NULL) {
         start = newnode;
         currentnode = newnode;
      } else {
         currentnode->next = newnode;
         newnode->prev = currentnode;
```

```
            currentnode = newnode;
        }

        printf("Press any key to make a new node or press 2 to exit: ");
        scanf("%d", &g);
        if (g == 2) {
            break;
        }
    }
}

void show() {
    struct node *ptr;
    ptr = start;
    while (ptr != NULL) {
        printf("%d -> ", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL\n");
}

void beforestart() {
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the data: \n");
    scanf("%d", &newnode->data);
    newnode->next = start;
    newnode->prev = NULL;

    if (start != NULL) {
```

```
        start->prev = newnode;
    }
    start = newnode;
}

void afterlast() {
    struct node *newnode, *ptr;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the data: \n");
    scanf("%d", &newnode->data);
    newnode->next = NULL;

    if (start == NULL) {
        start = newnode;
        newnode->prev = NULL;
    } else {
        ptr = start;
        while (ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = newnode;
        newnode->prev = ptr;
    }
}

void ascen() {
    struct node *newnode, *pre, *ptr;
    newnode = (struct node *)malloc(sizeof(struct node));
    printf("Enter the data: \n");
    scanf("%d", &newnode->data);
```

```
    newnode->next = NULL;
    newnode->prev = NULL;

    if (start == NULL || newnode->data < start->data) {
        newnode->next = start;
        if (start != NULL) {
            start->prev = newnode;
        }
        start = newnode;
    } else {
        ptr = start;
        while (ptr != NULL && newnode->data > ptr->data) {
            pre = ptr;
            ptr = ptr->next;
        }
        newnode->next = ptr;
        newnode->prev = pre;
        pre->next = newnode;
        if (ptr != NULL) {
            ptr->prev = newnode;
        }
    }
}

void delfn() {
    struct node *temp;
    if (start == NULL) {
        printf("There is nothing to delete\n");
    } else {
        temp = start;
```

```
        start = start->next;
        if (start != NULL) {
            start->prev = NULL;
        }
        free(temp);
    }
}

int main() {
    char ch;
    while (1) {
        printf("\nOptions:\n");
        printf("Press (m) to : Make the linked list.\n");
        printf("Press (s) to : Show the linked list.\n");
        printf("Press (a) to : Insert a node at the front of the linked list.\n");
        printf("Press (b) to : Insert a node at the end of the linked list.\n");
        printf("Press (c) to : Insert a node such that the linked list is in ascending
order.\n");
        printf("Press (d) to : Delete the first node of the linked list.\n");
        printf("Press (q) to : Exit the function.\n");

        scanf(" %c", &ch);
        switch (ch) {
            case 'm':
                make();
                break;
            case 's':
                show();
                break;
            case 'a':
```

```
            beforestart();
            break;
        case 'b':
            afterlast();
            break;
        case 'c':
            ascen();
            break;
        case 'd':
            delfn();
            break;
        case 'q':
            exit(0);
        default:
            printf("Invalid choice! Please try again.\n");
      }
   }
   return 0;
}
```

Output:

Nidhi Bisen
0801IT243D03 (B3-BATCH)

```
PS C:\Users\User\OneDrive\Desktop\DSA> gcc first.c
PS C:\Users\User\OneDrive\Desktop\DSA> ./a

Options:
Press (m) to : Make the linked list.
Press (s) to : Show the linked list.
Press (a) to : Insert a node at the front of the linked list.
Press (b) to : Insert a node at the end of the linked list.
Press (c) to : Insert a node such that the linked list is in as
cending order.
Press (d) to : Delete the first node of the linked list.
Press (q) to : Exit the function.
m
Enter the data:
12
Press any key to make a new node or press 2 to exit: 1
Enter the data:
32
Press any key to make a new node or press 2 to exit:
2

Options:
Press (m) to : Make the linked list.
Press (s) to : Show the linked list.
Press (a) to : Insert a node at the front of the linked list.
Press (b) to : Insert a node at the end of the linked list.
Press (c) to : Insert a node such that the linked list is in as
cending order.
Press (d) to : Delete the first node of the linked list.
Press (q) to : Exit the function.
s
12 -> 32 -> NULL

Options:
Press (m) to : Make the linked list.
Press (s) to : Show the linked list.
Press (a) to : Insert a node at the front of the linked list.
```

```
Press (b) to : Insert a node at the end of the linked list.
Press (c) to : Insert a node such that the linked list is in as
cending order.
Press (d) to : Delete the first node of the linked list.
Press (q) to : Exit the function.
c
Enter the data:
35

Options:
Press (m) to : Make the linked list.
Press (s) to : Show the linked list.
Press (a) to : Insert a node at the front of the linked list.
Press (b) to : Insert a node at the end of the linked list.
Press (c) to : Insert a node such that the linked list is in as
cending order.
Press (d) to : Delete the first node of the linked list.
Press (q) to : Exit the function.
s
12 -> 32 -> 35 -> NULL

Options:
Press (m) to : Make the linked list.
Press (s) to : Show the linked list.
Press (a) to : Insert a node at the front of the linked list.
Press (b) to : Insert a node at the end of the linked list.
Press (c) to : Insert a node such that the linked list is in as
cending order.
Press (d) to : Delete the first node of the linked list.
Press (q) to : Exit the function.
```