JADAVPUR UNIVERSITY

Department of Electronics and TeleCommunication Engineering

Data Structures and Algorithms Lab

Assignment - 2 : Stack(Fixed and Dynamic)

Programming Language : C (ANSI)

==

Sandip Dutta
2nd Year
Roll - 001910701017

==

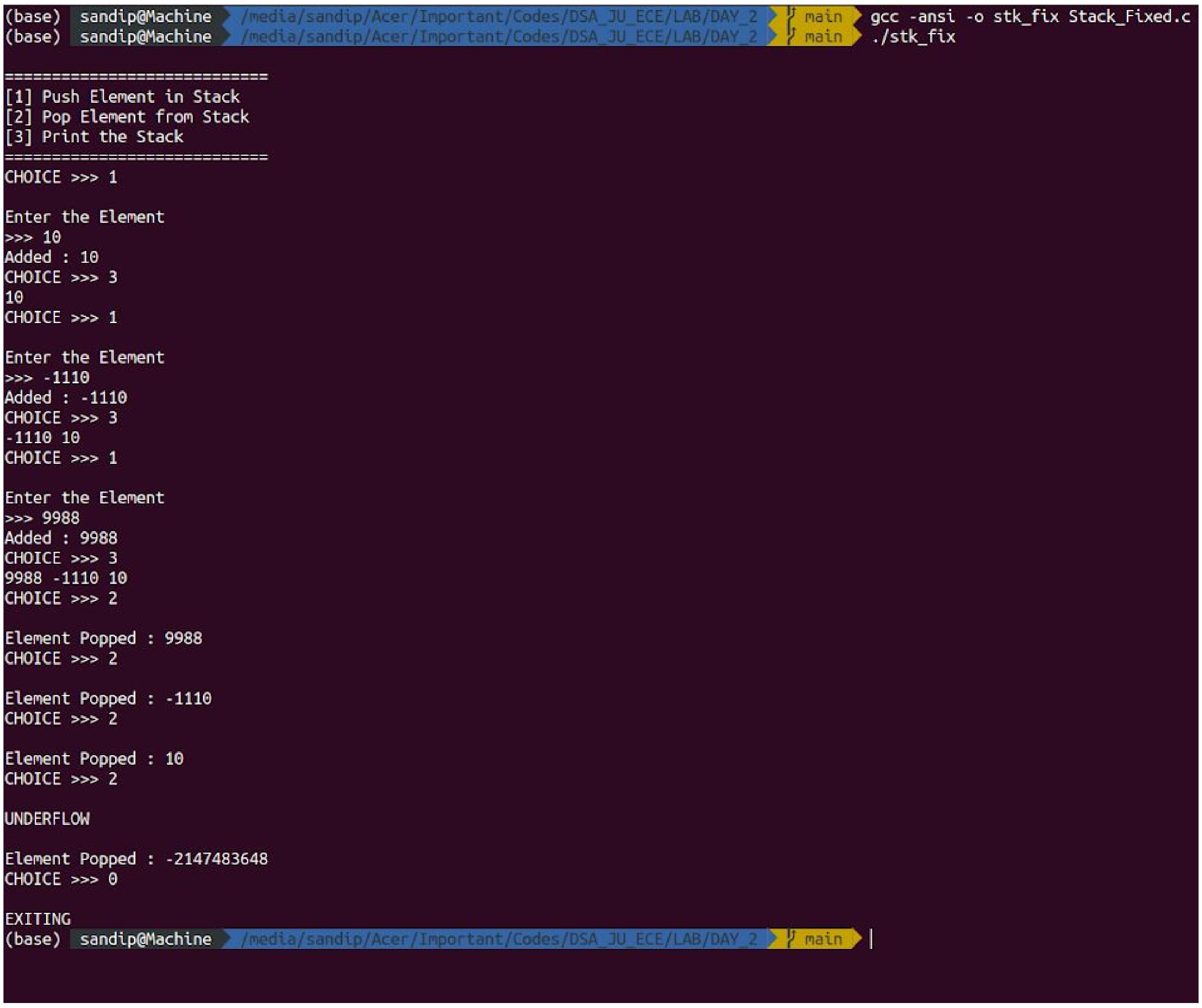
1. Stack - Fixed Memory Implementation

```
==============================================================================

/* Stack implementation with fixed memory array */
# include <stdio.h>
# include <stdlib.h>
# include <limits.h>
# define NULL_STACK -1
# define MAX_STACK_CAPACITY 100

/* Stack has a fixed memory capacity
a top pointer */
typedef struct stk{
    int capacity;
    int top;
    int* arr;
}Stack;

/* Init a stack */
Stack* initStack(const int capacity){
    Stack *stk = malloc(sizeof(Stack));

    stk -> top = NULL_STACK;
    stk -> capacity = capacity;
    stk -> arr = malloc(capacity * sizeof(int));

    if(stk == NULL || stk->arr == NULL){
        printf("\nNot enough memory! Aborting\n");
        return NULL;
    }

    return stk;
}

/* Checker method for empty or not */
int isEmpty(const Stack* stk){
    return (stk->top == NULL_STACK);
}

/* Checker to see if stack is full or not */
int isFull(const Stack* stk){
    return (stk->top == stk->capacity - 1);
}

/* Size of the stack */
int size(const Stack *stk){
    return stk->top + 1;
}

/* push data into stack */
void push(Stack *stk, int data){
    /* If full, OVERFLOW */
    if(isFull(stk)){
        printf("\nOVERFLOW\n");
        return;
    }
    printf("Added : %d\n", data);
    ++(stk -> top);
    stk -> arr[stk -> top] = data;
}

/* Pop data from stack */
int pop(Stack *stk){
    if(isEmpty(stk)){
        printf("\nUNDERFLOW\n");
        return INT_MIN;
    }
    return stk->arr[stk->top--];
}

/* print the stack */
void printStack(const Stack *stk){
    int lim = NULL_STACK;
    int i = stk -> top;

    if(isEmpty((stk))) return;
    while(i > lim){
        printf("%d ", stk->arr[i]);
        i--;
    }
    printf("\n");
}

void printMenu(){
    printf("\n============================\n");
    printf("[1] Push Element in Stack\n");
    printf("[2] Pop Element from Stack\n");
    printf("[3] Print the Stack\n");
    printf("============================\n");
}

int main(){
    int capacity = MAX_STACK_CAPACITY;
    Stack* stk = initStack(capacity);
    int element, choice;

    printMenu();

    do{
        printf("CHOICE >>> ");
        scanf("%d", &choice);

        switch(choice){
            default :
                printf("\nEXITING\n");
                choice = -1;
                break;
            case 1 :
                printf("\nEnter the Element\n>>> ");
                scanf("%d", &element);
                push(stk, element);
                break;
            case 2 :
                element = pop(stk);
                printf("\nElement Popped : %d\n", element);
                break;
            case 3:
                printStack(stk);
        }
    } while(choice != -1);
    return 0;
}
```

===

PTO

Output:



```
(base)  sandip@Machine  /media/sandip/Acer/Important/Codes/DSA_JU_ECE/LAB/DAY_2  main  gcc -ansi -o stk_fix Stack_Fixed.c
(base)  sandip@Machine  /media/sandip/Acer/Important/Codes/DSA_JU_ECE/LAB/DAY_2  main  ./stk_fix

============================
[1] Push Element in Stack
[2] Pop Element from Stack
[3] Print the Stack
============================
CHOICE >>> 1

Enter the Element
>>> 10
Added : 10
CHOICE >>> 3
10
CHOICE >>> 1

Enter the Element
>>> -1110
Added : -1110
CHOICE >>> 3
-1110 10
CHOICE >>> 1

Enter the Element
>>> 9988
Added : 9988
CHOICE >>> 3
9988 -1110 10
CHOICE >>> 2

Element Popped : 9988
CHOICE >>> 2

Element Popped : -1110
CHOICE >>> 2

Element Popped : 10
CHOICE >>> 2

UNDERFLOW

Element Popped : -2147483648
CHOICE >>> 0

EXITING
(base)  sandip@Machine  /media/sandip/Acer/Important/Codes/DSA_JU_ECE/LAB/DAY_2  main  |
```

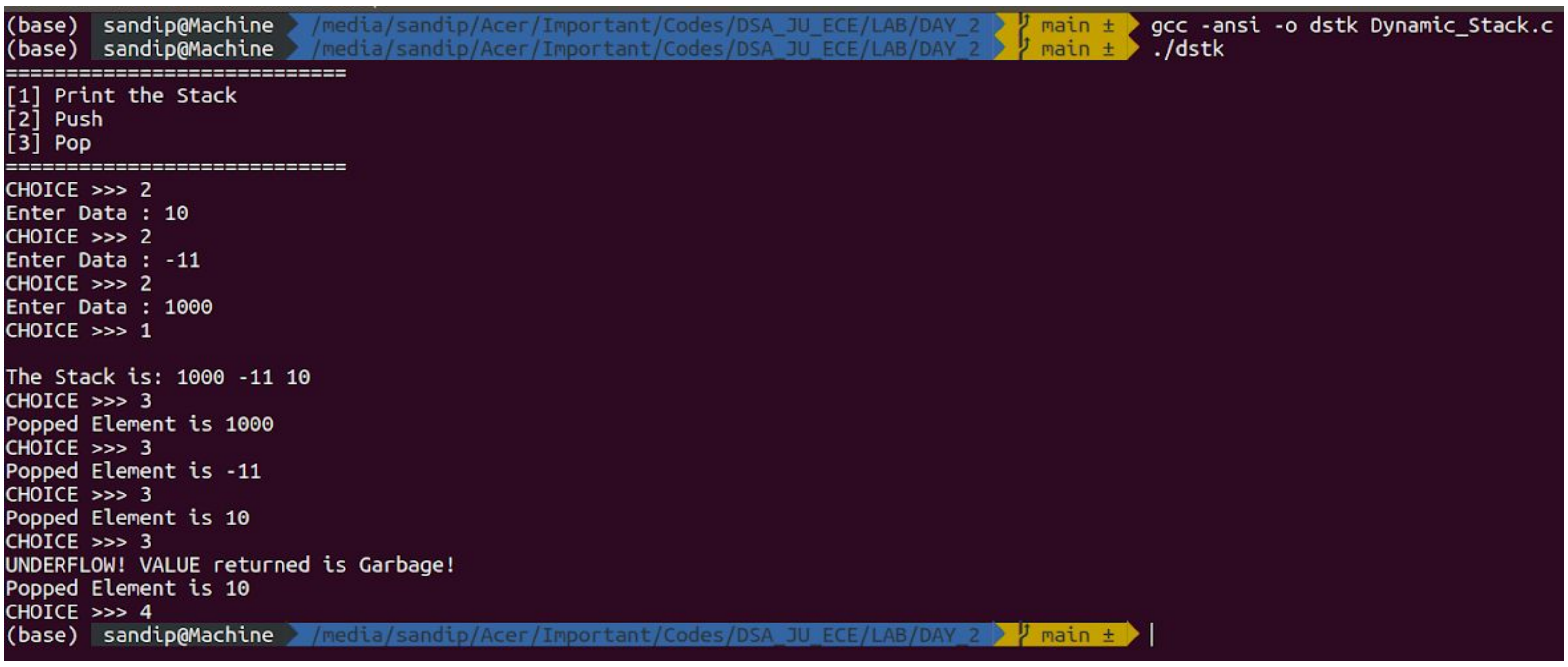
2. Dynamic Memory Stack using Linked List (Singly Connected)

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

/* Dynamic Stack Implementation using Linked List */
/* Node*/
struct SLLNode{
    int data;
    struct SLLNode *next;
};

/*Length of Linked List*/
int get_length(struct SLLNode *head){
    struct SLLNode *curr = head;
    int length = 1;

    if (head == NULL) return 0;
    while(curr = curr -> next) ++length;
    return length;
}

/* Print the list */
void printList(struct SLLNode *head){
    if(head == NULL) return;
    struct SLLNode *curr = head;

    while(curr) {
        printf("%d ", curr -> data);
        curr = curr -> next;
    }
    printf("\n");
}

/* Insert in SLL*/
struct SLLNode* push(int data, struct SLLNode* head){
    struct SLLNode *newNode;

    newNode = (struct SLLNode*) malloc(sizeof(struct SLLNode));
    newNode -> data = data;
    if(!newNode) return NULL;
    newNode -> next = head;
    return newNode;
}

/* Pop from SLL */
struct SLLNode* pop(struct SLLNode *head, int *element){
    /* element is used for returning popped value */
    struct SLLNode *p;
    if(head == NULL) {
        printf("UNDERFLOW! VALUE returned is Garbage!\n");
        return head;
    }
    p = head;
    head = head -> next;
    *element = p -> data;
    free(p);
    return head;
}

/*Show menu*/
void show_menu(){
    printf("============================\n");
    printf("[1] Print the Stack\n");
    printf("[2] Push\n");
    printf("[3] Pop\n");
    printf("============================\n");
}
```

```
/* Driver Function */
int main(){
    struct SLLNode *head = NULL;
    int choice, data, element;

    show_menu();

    do{
        printf("CHOICE >>> ");
        scanf("%d", &choice);

        switch (choice){
            case 1:
                printf("\nThe Stack is: ");
                printList(head);
                break;
            case 2:
                printf("Enter Data : ");
                scanf("%d", &data);
                head = push(data, head);
                break;
            case 3:
                head = pop(head, &element);
                printf("Popped Element is %d\n", element);
                break;
            default:
                choice = -1;
                break;
        }
    }while(choice != -1);
    return 0;
}
```

===

Output



```
(base)  sandip@Machine  /media/sandip/Acer/Important/Codes/DSA_JU_ECE/LAB/DAY_2  main ±  gcc -ansi -o dstk Dynamic_Stack.c
(base)  sandip@Machine  /media/sandip/Acer/Important/Codes/DSA_JU_ECE/LAB/DAY_2  main ±  ./dstk
===========================
[1] Print the Stack
[2] Push
[3] Pop
===========================
CHOICE >>> 2
Enter Data : 10
CHOICE >>> 2
Enter Data : -11
CHOICE >>> 2
Enter Data : 1000
CHOICE >>> 1

The Stack is: 1000 -11 10
CHOICE >>> 3
Popped Element is 1000
CHOICE >>> 3
Popped Element is -11
CHOICE >>> 3
Popped Element is 10
CHOICE >>> 3
UNDERFLOW! VALUE returned is Garbage!
Popped Element is 10
CHOICE >>> 4
(base)  sandip@Machine  /media/sandip/Acer/Important/Codes/DSA_JU_ECE/LAB/DAY_2  main ±  |
```