DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMMING LABORATORY (CSE 351)

ASSIGNMENT 6

Asmit De
10/CSE/53

Date: 20.10.2011

Program 1: Selection Sort

Source Code –

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

void SelectionSort(int *arr, int n)
{
	int i, j, k, t;

	for(i = 0; i <= n-2; i++)
	{
		k = i;
		for(j = i+1; j <= n-1; j++)
		{
			if(arr[j] > arr[k])
				k = j;
		}

		t = arr[i];
		arr[i] = arr[k];
		arr[k] = t;
	}
}

int main()
{

	int i, n, *arr;
	system("cls");

	printf("Enter the number of elements to insert: ");
	scanf("%d", &n);

	arr = (int*)malloc(n*sizeof(int));

	for(i=0; i<=n-1; i++)
	{
		printf("Enter #%d: ", i);
		scanf("%d", &arr[i]);
	}

	SelectionSort(arr, n);

	printf("\nSorted array in descending order:\n");
	for(i=0; i<=n-1; i++)
		printf("%d\n", arr[i]);

	getch();
	return 0;
}
```

Output–

```
Enter the number of elements to insert: 5
Enter #0: 34
Enter #1: 12
Enter #2: 86
Enter #3: 45
Enter #4: 55

Sorted array in descending order:
86
55
45
34
12
```

Program 2: Insertion Sort

Source Code –

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

void InsertionSort(int *arr, int n)
{
	int i, j, key;

	for(i = 1; i <= n-1; i++)
	{
		key = arr[i];
		j = i-1;
		while(arr[j] > key && j >= 0)
		{
			arr[j+1] = arr[j];
			j--;
		}
		arr[j+1] = key;
	}
}

int main()
{

	int i, n, *arr;
	system("cls");

	printf("Enter the number of elements to insert: ");
	scanf("%d", &n);

	arr = (int*)malloc(n*sizeof(int));

	for(i = 0; i <= n-1; i++)
	{
		printf("Enter #%d: ", i);
		scanf("%d", &arr[i]);
	}

	InsertionSort(arr, n);

	printf("\nSorted array in ascending order:\n");
	for(i=0; i<=n-1; i++)
		printf("%d\n", arr[i]);

	getch();
	return 0;
}
```

Output –

```
Enter the number of elements to insert: 5
Enter #0: 34
Enter #1: 98
Enter #2: 12
Enter #3: 55
Enter #4: 46

Sorted array in ascending order:
12
34
46
55
98
```

Program 3: Implementation of a Queue using Array

Source Code –

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define SIZE 100

typedef struct Queue
{
	int _queue[SIZE];
	int front, rear;
}Queue;

void initQueue(Queue *q)
{
	q->front = -1;
	q->rear = -1;
}

int isFull(Queue *q)
{
	if(q->rear == SIZE-1)
		return 1;
	else
		return 0;
}

int isEmpty(Queue *q)
{
	if(q->front == q->rear)
		return 1;
	else
		return 0;
}

int enqueue(Queue *q, int data)
{
	if(isFull(q))
	{
		printf("\n\aError: Queue is full...");
		return 1;
	}

	q->_queue[++q->rear] = data;
	return 0;
}

int dequeue(Queue *q, int *data)
{
	if(isEmpty(q))
	{
		printf("\n\aError: Queue is empty...");
		return 1;
	}

	*data = q->_queue[++q->front];
	return 0;
}

void displayQueue(Queue *q)
{
	int i;
	printf("Queue: ");
	for(i = q->rear; i >= q->front+1; i--)
		printf("%d ", q->_queue[i]);
}
```

```
int main()
{
	Queue *queue = (Queue*)malloc(sizeof(Queue));
	int *data = (int*)malloc(sizeof(int));
	char choice;

	initQueue(queue);
	while(1)
	{
		system("cls");
		puts("MENU");
		puts("\nKey \tFunction");
		puts("1 \tDisplay Queue");
		puts("2 \tEnqueue Data");
		puts("3 \tDequeue Data");
		puts("4 \tClear Queue");
		puts("X \tExit");
		printf("\nEnter choice...");
		choice = getch();
		fflush(stdin);

		switch(choice)
		{
		case '1':
			system("cls");

			displayQueue(queue);

			printf("\n\nPress any key to return to menu...");
			getch();
			break;

		case '2':
			system("cls");

			printf("Enter data: ");
			scanf("%d", data);
			if(!enqueue(queue, *data))
				printf("\nData queued successfully...");

			printf("\n\nPress any key to return to menu...");
			getch();
			break;

		case '3':
			system("cls");

			if(!dequeue(queue, data))
				printf("\nData retrieved: %d", *data);

			printf("\n\nPress any key to return to menu...");
			getch();
			break;

		case '4':
			system("cls");

			initQueue(queue);
			printf("Queue cleared successfully...");

			printf("\n\nPress any key to return to menu...");
			getch();
			break;

		case 'X':
		case 'x':
			exit(0);
```

```
            default:
                   system("cls");

                   printf("\aError: Invalid Input...");

                   printf("\n\nPress any key to return to menu...");
                   getch();
            }
     }
}
```

Program 4: Implementation of a Queue using Linked List

Source Code –

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

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

NODE *getNode(int data)
{
	NODE *node = (NODE*)malloc(sizeof(NODE));

	if(node == NULL)
		return NULL;

	node->data = data;
	node->next = NULL;
	return node;
}

NODE *enqueue(NODE *list, int data)
{
	NODE *node = getNode(data);

	if(node == NULL)
	{
		printf("\n\aError: Queue is full...");
		return NULL;
	}

	node->next = list;
	return node;
}

NODE *dequeue(NODE *list, int *data)
{
	NODE *temp = list;

	if(list->next == NULL)
	{
		*data = list->data;
		free(list);
		return NULL;
	}
	else
	{
		while(list->next->next != NULL)
			list = list->next;

		*data = list->next->data;
		free(list->next);
		list->next = NULL;
		return temp;
	}
}

void displayQueue(NODE *list)
{
	printf("\nQueue: ");
	while(list != NULL)
	{
		printf("%d ", list->data);
```

```
            list = list->next;
        }
}

NODE *clearQueue(NODE *list)
{
        NODE *temp = NULL;

        while(list != NULL)
        {
                temp = list;
                list = list->next;
                free(temp);
        }
        return list;
}

int main()
{
        NODE *queue = NULL;
        int *data = (int*)malloc(sizeof(int));
        char choice;

        while(1)
        {
                system("cls");
                puts("MENU");
                puts("\nKey \tFunction");
                puts("1 \tDisplay Queue");
                puts("2 \tEnqueue Data");
                puts("3 \tDequeue Data");
                puts("4 \tClear Queue");
                puts("X \tExit");
                printf("\nEnter choice...");
                choice = getch();
                fflush(stdin);

                switch(choice)
                {
                case '1':
                        system("cls");

                        displayQueue(queue);

                        printf("\n\nPress any key to return to menu...");
                        getch();
                        break;

                case '2':
                        system("cls");

                        printf("Enter data: ");
                        scanf("%d", data);
                        queue = enqueue(queue, *data);
                        if(queue != NULL)
                                printf("\nData queued successfully...");

                        printf("\n\nPress any key to return to menu...");
                        getch();
                        break;

                case '3':
                        system("cls");

                        if(queue != NULL)
                        {
                                queue = dequeue(queue, data);
                                printf("\nData retrieved: %d", *data);
                        }
                        else
                                printf("\n\aError: Queue is empty...");

                        printf("\n\nPress any key to return to menu...");
                        getch();
                        break;

                case '4':
                        system("cls");

                        queue = clearQueue(queue);
                        printf("Queue cleared successfully...");

                        printf("\n\nPress any key to return to menu...");
                        getch();
                        break;

                case 'X':
                case 'x':
                        exit(0);

                default:
                        system("cls");

                        printf("\aError: Invalid Input...");

                        printf("\n\nPress any key to return to menu...");
                        getch();
                }
        }
}
```